(3)437



WL-TR-92-7044

# Numerical Solution and Algorithm Analysis for the Unsteady Navier-Stokes Equations on Dynamic Multiblock Grids Volume I

J. M. Janus
A. Arabshahi
D. L. Whitfield

Mississippi State University
P.O. Box 6176
Mississippi State MS 39762

OCTOBER 1992

FINAL REPORT FOR PERIOD JUNE 1989 - JUNE 1992



Approved for public release; distribution is unlimited.



WRIGHT LABORATORY, ARMAMENT DIRECTORATE
Air Force Materiel Command ▮ United States Air Force ▮ Eglin Air Force Base

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise as in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

Robert F. Donohue Jr.

ROBERT F. DONOHUE, Jr.
Chief, Weapon Flight Mechanics Division

Even though this report may contain special release rights held by the controlling office, please do not request copies from the Wright Laboratory, Armament Directorate. If you qualify as a recipient, release approval will be obtained from the originating activity by DTIC. Address your request for additional copies to:

Defense Technical Information Center
Cameron Station
Alexandria VA 22304-6145

If your address has changed, if you wish to be removed from our mailing list, or if your organization no longer employs the addressee, please notify WL/MN AA , Eglin AFB FL 32542-5000, to help us maintain a current mailing list.

Do not return copies of this report unless contractual obligations or notice on a specific document requires that it be returned.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | October 1992 | Final June 1989 - June 1992 |

**4. TITLE AND SUBTITLE**
Numerical Solution and Algorithm Analysis for the Unsteady Navier-Stokes Equations on Dynamic Multiblock Grids

**5. FUNDING NUMBERS**
C: F08635-89-C-0208
PE: 61102F
PR: 2307
TA: AW
WU: 15

**6. AUTHOR(S)**
J. M. Janus, A. Arabshahi, and D. L. Whitfield

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Mississippi State University
P.O. Box 6176
Mississippi State MS 39762

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Wright Laboratory, Armament Directorate
Weapon Flight Mechanics Division
Aerodynamics Branch (WL/MNAA)
Eglin AFB FL 32542-5000

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
WL-TR-92-7044
Volume I

**11. SUPPLEMENTARY NOTES**
Due to the enormous backlog of work accumulated because of the DMR manpower reduction, and in the interest of getting this technology out to the users, **the text was not edited by TESCO, Inc.**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**12b. DISTRIBUTION CODE**
A

**13. ABSTRACT** *(Maximum 200 words)*
This is a two volume final report for the research entitled Computation of Hypersonic Interference Flowfields. This volume involves the numerical solution of the three-dimensional unsteady Euler and Navier-Stokes equations for a calorically perfect gas. Volume II involves the development techniques for the solution of three-dimensional inviscid and viscous hypersonic flows in chemical equilibrium. This report addresses the numerical formulation and solution of the equations as well as an analysis of the optimization of the numerical solution algorithm used to solve the equations. Selected results of numerious hypersonic flow computations--including computations of heat transfer to bodies with different nose shapes--are presented.

**14. SUBJECT TERMS**
Store Separation
Computational Fluid Dynamics
Navier-Stokes Equations
Euler Equations

**15. NUMBER OF PAGES**

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

## PREFACE

This program was conducted by Mississippi State University, P.O. Box 6176, Mississippi State MS 39762, under Contract No. F08635-89-C-0208 with the Wright Laboratory, Armament Directorate, Eglin Air Force Base FL 32542-5000. Dr. Dave M. Belk, WL/MNAA, managed the program for the Wright Laboratory. The principal investigators were Drs. J. Mark Janus and David L. Whitfield of Mississippi State University. The program was conducted from 12 June 1989 through 11 June 1992.

TABLE OF CONTENTS

LIST OF FIGURES

## LIST OF FIGURES (Concluded)

## LIST OF SYMBOLS

| | |
|---|---|
| $a$ | speed of sound |
| $A, B, C$ | Jacobian of the flux vectors F, G, H |
| $C_{Kleb}$ | Klebanoff intermittency constant |
| $D$ | left hand side matrix of the modified two–pass method |
| $e$ | total energy per unit volume |
| $e_i$ | internal energy per unit mass |
| $f, g, h$ | Cartesian coordinate flux vectors |
| $F, G, H$ | curvilinear coordinate flux vectors |
| $F_{Kleb}$ | Klebanoff intermittency factor |
| $F_{wake}$ | wake function in turbulence model |
| $I$ | identity matrix |
| $J$ | Jacobian of coordinate transformation, $\det\left\|\frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)}\right\|$ |
| $K$ | thermal conductivity; Clauser constant |
| $k$ | von Karman constant |
| $l$ | mixing length |
| $L$ | reference length |
| $M$ | sum of flux Jacobians in each direction; Mach number |
| $p$ | pressure |
| $Pr$ | Prandtl number |
| $q$ | Cartesian coordinate conservative variable vector |
| $q_x, q_y, q_z$ | heat flux terms in x, y, z directions |
| $Q$ | curvilinear coordinate conservative variable vector |
| $R$ | residual |
| $Re$ | Reynolds number |
| $t$ | time |
| $T$ | temperature |

## LIST OF SYMBOLS (Continued)

| | |
|---|---|
| u, v, w | Cartesian components of the absolute velocity vector |
| U, V, W | contravariant velocities |
| $\vec{V}$ | velocity vector |
| x, y, z | Cartesian coordinates |
| X | solution of vector of matrix inversion |
| Y | normal distance of a point to the wall |
| $Y^+$ | nondimensional length in turbulence model |
| $\gamma$ | ratio of specific heats, $C_p/C_v$ |
| $\lambda$ | second coefficient of viscosity |
| $\delta$ | central difference operator |
| $\mu$ | viscosity |
| $\mu_\tau$ | turbulent eddy viscosity |
| $\varrho$ | density |
| $\tau$ | time in computational space; fluid stress vector |
| $\Delta\tau$ | time step |
| $\omega$ | vorticity, $\nabla \times \vec{V}$ |
| $\xi, \eta, \varsigma$ | curvilinear coordinates |

## LIST OF SYMBOLS (Concluded)

Subscripts

| | |
|---|---|
| f | field point |
| i, j, k | in the $\xi$, $\eta$, $\varsigma$ direction |
| o | total condition |
| t | time differentiation turbulent |
| v | diffusive quantity |
| w | wall |
| x, y, z | partial differentiation |
| $\xi$, $\eta$, $\varsigma$ | partial differentiation |
| $\infty$ | freestream reference value |

Superscripts

| | |
|---|---|
| –1 | inverse |
| n | previous time step |
| n+1 | current time step |
| + | corresponding to positive eigenvalues |
| – | corresponding to negative eignevalues |

# SECTION I

# INTRODUCTION

This is the final report for research in the area of Computation of Hypersonic Interference Flowfields for the U.S. Air Force. Basic guidelines for the flow regime of concern was a free-stream Mach number of 10 and less at an altitude of 100,000 feet and below. A significant portion of this research was associated with solving hypersonic flow problems in chemical equilibrium. This hypersonic chemical equilibrium flow effort was sufficiently large and self–contained that a separate report was devoted to this task. This report is entitled "An Efficient Solver for Flows in Local Chemical Equilibrium" and it will be published as an AFATL report.

This report contains the equations solved, the numerical cell face flux formulation used, the numerical methods developed for solving the system of discretized equations, a discussion of software optimization, and example results. The equations solved are the compressible form of the three–dimensional unsteady Euler and Navier–Stokes equations, and these equations are discussed in Section II. The numerical flux formulation used here is neither a Monotone Upstream–centered Schemes for Conservation Laws (MUSCL) approach, which is a dependent variable extrapolation method, nor a flux extrapolation method; rather it falls somewhere in between. The development of the present numerical flux formulation and how it differs from the dependent variable extrapolation and flux extrapolation methods is presented in Section III.

During the course of this research various numerical solution schemes were developed and applied to the solution of the equations for both steady and unsteady flow. This was a rather important aspect of the research because it was discovered that the previously used numerical solution scheme, the so–called two–pass scheme, had difficulties in solving the equations on grids that were clustered on both ends of a computational coordinate direction. For simple external flow problems where the grid is typically clustered near the the body and stretched as the distance from the body increases, the old two–pass scheme seemed to work rather well. But for a situation where the grid was clustered on both ends of a computational coordinate such as might occur along a grid line that runs from a store to the fuselage, for example, numerical difficulties were encountered. This was particularly noticeable for Navier–Stokes grids where the clustering could be severe. A new scheme, referred to as the modified two–pass scheme, has provided a method for solving such problems for all cases considered thus far. In addition, an extension of this modified two–pass scheme was developed and it is referred to as the N–pass scheme. A review of the old two–pass scheme and the development of these new numerical solution schemes are presented in Section IV.

The numerical solution schemes presented in Section IV have developed into rather workhorse methods for solving the three–dimensional Euler and Navier–Stokes equations for reasonably complex configurations. Because of the collective amount of computer resources being devoted to solving such problems, an effort was directed toward investigating ways of improving the computational efficiency of these numerical solution schemes. Results of this effort are presented in Section V.

Numerous computations have been performed using the computational techniques discussed in this report. Selected examples of the various results are given in Section VI. Concluding remarks are given in Section VII.

## SECTION II

## GOVERNING EQUATIONS

### 1. NAVIER–STOKES EQUATIONS

The equations of motion which describe the behavior of a continuum fluid flow are the Navier–Stokes equations. In the absence of body forces and heat sources, the conservation laws for mass, momentum, and energy of a perfect gas over a stationary finite volume $\Omega$, enclosed by a control surface are expressed by the following integral form of the non–dimensional Navier–Stokes equations

$$\frac{\partial}{\partial t}\int\int\int_{\Omega} \vec{q}dv + \int\int_{s} \tilde{F}(\vec{q}) \cdot \vec{n}ds + \int\int_{s} \tilde{F}_v(\vec{q}, \nabla\vec{q}) \cdot \vec{n}ds = 0 \qquad (1)$$

where

$$\vec{q} = [\ \varrho, \varrho u, \varrho v, \varrho w, e]^T$$

$$\tilde{F} = [\ \bar{f}, \bar{g}, \bar{h}\ ]^T$$

$$\tilde{F}_v = [\ \bar{f}_v, \bar{g}_v, \bar{h}_v\ ]^T$$

The convected flux vectors are

$$\bar{f} = [\ \varrho u\ ,\ \varrho u^2 + p\ ,\ \varrho uv\ ,\ \varrho uw\ ,\ u(e + p)\ ]^T$$

$$\bar{g} = [\ \varrho v\ ,\ \varrho uv\ ,\ \varrho v^2 + p\ ,\ \varrho vw\ ,\ v(e + p)\ ]^T$$

$$\bar{h} = [\ \varrho w\ ,\ \varrho uw\ ,\ \varrho vw\ ,\ \varrho w^2 + p\ ,\ w(e + p)\ ]^T$$

$$e = e_i + \frac{1}{2}\varrho\ (u^2 + v^2 + w^2)$$

The vectors of diffusive terms are

$$\bar{f}_v = \frac{M_\infty}{Re_L}\ [\ 0\ ,\ \tau_{xx}\ ,\ \tau_{xy}\ ,\ \tau_{xz}\ ,\ \tau_{xx}u + \tau_{xy}v + \tau_{xz}w - q_x\ ]^T$$

$$\bar{g}_v = \frac{M_\infty}{Re_L}\ [\ 0\ ,\ \tau_{yx}\ ,\ \tau_{yy}\ ,\ \tau_{yz}\ ,\ \tau_{yx}u + \tau_{yy}v + \tau_{yz}w - q_y\ ]^T$$

$$\bar{h}_v = \frac{M_\infty}{Re_L}\ [\ 0\ ,\ \tau_{zx}\ ,\ \tau_{zy}\ ,\ \tau_{zz}\ ,\ \tau_{zx}u + \tau_{zy}v + \tau_{zz}w - q_z\ ]^T$$

The viscous stress and heat flux terms are

$$\tau_{xx} = (2\mu + \lambda)\, u_x + \lambda(v_y + w_z)$$

$$\tau_{yy} = (2\mu + \lambda)\, v_y + \lambda(u_x + w_z)$$

$$\tau_{zz} = (2\mu + \lambda)\, w_z + \lambda(u_x + v_y)$$

$$\tau_{xy} = \tau_{yx} = \mu\,(u_y + v_x)$$

$$\tau_{xz} = \tau_{zx} = \mu\,(u_z + w_x)$$

$$\tau_{yz} = \tau_{zy} = \mu\,(v_z + w_y)$$

$$q_x = -K\frac{\partial T}{\partial x}$$

$$q_y = -K\frac{\partial T}{\partial y} \qquad \text{and} \qquad K = \frac{\mu}{(\gamma - 1)\,Pr}$$

$$q_z = -K\frac{\partial T}{\partial z}$$

where K is the thermal conductivity, and T is the temperature.

Here conventional definitions of the flow quantities are used. The Cartesian velocities components (u,v,w) are normalized by the free–stream speed of sound $a_\infty$, $\varrho$ is normalized by $\varrho_\infty$, the internal energy $e_i$, the total energy e and the pressure p are normalized by $\varrho_\infty\, a_\infty^2$. The two viscosity coefficients $\lambda$ and $\mu$ are normalized by the molecular viscosity $\mu_\infty$. The constant $\gamma$ is the ratio of specific heats, $Re_L$ is the Reynolds number based on free–stream velocity and reference length L, and Pr is the Prandtl number. For a perfect gas the normalized state relations are

$$p = \frac{1}{\gamma}\,\varrho\, T\,, \quad e_i = \frac{T}{\gamma(\gamma - 1)}\,, \quad a^2 = T$$

where the temperature T is normalized with respect to $T_\infty$. The system of equations (Equation 1) is valid for turbulent as well as laminar flows by replacing the molecular transport coefficients with their turbulent counterparts. The governing equations become the so–called Reynold–averaged Navier–Stokes equations.

For high–Reynolds–numbers flows, the viscous effects are confined to a thin layer near the wall boundary and dominated by the viscous terms associated with the strain rates normal to the wall. The viscous terms associated with the strain rates along the body surface are comparatively small and negligible. This concept was first discussed by Prandtl in the development of boundary–layer theory and has been applied and extended to various problems. The development of the single thin–layer approximation of the full Navier–Stokes equations was introduced

by Steger [Reference 1] and used by Gatlin [Reference 2] and Simpson [Reference 3], in which only viscous terms in the body–normal, $\eta$, direction are retained. Here the concept is extended to the case of thin–layer approximation to two directions for a general coordinate system. All the viscous terms associated with cross–derivatives where ($\partial^2/\partial x^i\ \partial x^j$ where $i \neq j$) are neglected, but with retention of the viscous terms with normal second derivatives ($\partial^2/\partial x^i\ \partial x^j$ where $i = j$). This approximation retains the most dominant terms in the governing equation. The thin–layer Navier–Stokes equations are obtained by retaining from the full Navier–Stokes equations all the viscous terms except for those containing derivatives in the streamwise, $\xi$, direction, as well as excluding all cross–derivatives.

The time–dependent thin–layer Navier–Stokes equations in general body–fitted coordinates, written in nondimensional strong conversation law form are

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F}{\partial \xi} + \frac{\partial (G - G_v)}{\partial \eta} + \frac{\partial (H - H_v)}{\partial \zeta} = 0 \qquad (2)$$

where the curvilinear coordinates are defined as

$$\xi = \xi(x, y, z, t) \quad , \quad \zeta = \zeta(x, y, z, t)$$
$$\eta = \eta(x, y, z, t) \quad , \quad \tau = t$$

and the vector of the dependent variable Q and the Euler flux vector F, G, and H are given by

$$Q = J \begin{bmatrix} \varrho \\ \varrho u \\ \varrho v \\ \varrho w \\ e \end{bmatrix} \quad , \quad F = J \begin{bmatrix} \varrho U \\ \varrho u U + \xi_x p \\ \varrho v U + \xi_y p \\ \varrho w U + \xi_z p \\ U(e + p) - \xi_t p \end{bmatrix}$$

$$G = J \begin{bmatrix} \varrho V \\ \varrho u V + \eta_x p \\ \varrho v V + \eta_y p \\ \varrho w V + \eta_z p \\ V(e + p) - \eta_t p \end{bmatrix} \quad , \quad H = J \begin{bmatrix} \varrho W \\ \varrho u W + \zeta_x p \\ \varrho v W + \zeta_y p \\ \varrho w W + \zeta_z p \\ W(e + p) - \zeta_t p \end{bmatrix}$$

with the contravariant velocities, U, V, and W, defined as

$$U = \xi_t + \xi_x u + \xi_y v + \xi_z w$$
$$V = \eta_t + \eta_x u + \eta_y v + \eta_z w$$
$$W = \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w$$

and the quantity J is the Jacobian of the inverse transformation and is given by

$$J = x_\xi(y_\eta z_\zeta - z_\eta y_\zeta) - y_\xi(x_\eta z_\zeta - z_\eta x_\zeta) + z_\xi(x_\eta y_\zeta - y_\eta x_\zeta)$$

The metric quantities are given by

$$\begin{aligned}
\xi_x &= J^{-1}(y_\eta z_\zeta - z_\eta y_\zeta) & \eta_x &= J^{-1}(z_\xi y_\zeta - y_\xi z_\zeta) \\
\xi_y &= J^{-1}(z_\eta x_\zeta - x_\eta z_\zeta) & \eta_y &= J^{-1}(x_\xi z_\zeta - z_\xi x_\zeta) \\
\xi_z &= J^{-1}(x_\eta y_\zeta - y_\eta x_\zeta) & \eta_z &= J^{-1}(x_\zeta y_\xi - y_\zeta x_\xi) \\
\zeta_x &= J^{-1}(y_\xi z_\eta - z_\xi y_\eta) & \xi_t &= (-x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z) \\
\zeta_y &= J^{-1}(x_\eta z_\xi - z_\eta x_\xi) & \eta_t &= (-x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z) \\
\zeta_z &= J^{-1}(x_\xi y_\eta - y_\xi x_\eta) & \zeta_t &= (-x_\tau \zeta_x - y_\tau \zeta_y - z_\tau \zeta_z)
\end{aligned}$$

The pressure, density, and velocity components are related to the energy for an ideal gas by the following equation

$$p = (\gamma - 1)\left[e - \frac{1}{2}\varrho(u^2 + v^2 + w^2)\right]$$

where the ratio of specific heats, $\gamma$, is taken as 1.4

The general form of the thin–layer viscous flux vector can be written as

$$S_v = J\begin{bmatrix} 0 \\ T_{kx} \\ T_{ky} \\ T_{kz} \\ uT_{kx} + vT_{ky} + wT_{kz} - \Gamma_k \end{bmatrix} \tag{3}$$

where

$$\begin{aligned}
T_{kx} &= k_x\tau_{xx} + k_y\tau_{xy} + k_z\tau_{xz} \\
T_{ky} &= k_x\tau_{xy} + k_y\tau_{yy} + k_z\tau_{yz} \\
T_{kz} &= k_x\tau_{xz} + k_y\tau_{yz} + k_z\tau_{zz} \\
\Gamma_k &= k_x q_x + k_y q_y + k_z q_z
\end{aligned}$$

The viscous flux vectors $G_v$ and $H_v$ are given by the $S_v$ vector depending on whether k in Equation (3) is $\eta$, or $\varsigma$, respectively.

The elements of the viscous stress tensor are

$$
\begin{aligned}
\tau_{xx} &= \frac{2\mu M_\infty}{3\,Re_L}\left[2k_x u_k - k_y v_k - k_z w_k\right] \\
\tau_{yy} &= \frac{2\mu M_\infty}{3\,Re_L}\left[2k_y v_k - k_x u_k - k_z w_k\right] \\
\tau_{zz} &= \frac{2\mu M_\infty}{3\,Re_L}\left[2k_z w_k - k_x u_k - k_y v_k\right] \\
\tau_{xy} &= \tau_{yx} = \frac{\mu M_\infty}{Re_L}\left[k_y u_k + k_x v_k\right] \\
\tau_{xz} &= \tau_{zx} = \frac{\mu M_\infty}{Re_L}\left[k_z u_k + k_x w_k\right] \\
\tau_{yz} &= \tau_{zy} = \frac{\mu M_\infty}{Re_L}\left[k_z v_k + k_y w_k\right]
\end{aligned}
\tag{4}
$$

and the heat flux are

$$
\begin{aligned}
q_x &= \frac{-\mu M_\infty}{(\gamma-1)\,Pr\,Re_L}\,k_x T_k \\
q_y &= \frac{-\mu M_\infty}{(\gamma-1)\,Pr\,Re_L}\,k_y T_k \\
q_z &= \frac{-\mu M_\infty}{(\gamma-1)\,Pr\,Re_L}\,k_z T_k
\end{aligned}
\tag{5}
$$

The element of the viscous stress tensor, Equation (4), and the heat flux terms, Equation (5), belong to the viscous flux vectors $G_v$, or $H_v$, when k in these equations is $\eta$, or $\varsigma$, respectively.

A finite volume method is adopted to ensure the final converged solution is independent of the integration procedure and to avoid metric singularity problems. An implicit discretized integral form of the thin–layer Navier–Stokes equations, Equation (2), in computation space for a cell with center denoted as (i, j, k) is

$$
\Delta Q^n + \Delta\tau\left[\delta_i F^{n+1} + \delta_j\,(G - G_v)^{n+1} + \delta_k\,(H - H_v)^{n+1}\right] = 0 \tag{6}
$$

where

$$
\begin{aligned}
\Delta Q^n &= Q^{n+1} - Q^n \\
\delta_*(\cdot) &= (\cdot)_{*+\frac{1}{2}} - (\cdot)_{*-\frac{1}{2}}
\end{aligned}
$$

and i, j, and k are the indices in the $\xi$, $\eta$, and $\varsigma$ direction, respectively, and n is the temporal index. In this equation, the dependent variable Q is considered to be constant throughout cell (i, j,

k), while the inviscid and viscous fluxes are assumed to be uniform over each of the six surfaces of the cell. An unfactored implicit scheme can be obtained from the Equation (6) by linearizing the inviscid and viscous flux vectors about the previous time level and dropping terms of the second or higher order, resulting in the following linear equation

$$\left[I + \Delta\tau \left(\delta_i A \cdot + \delta_j B \cdot + \delta_k C \cdot - \delta_j B_v \cdot - \delta_k C_v \cdot \right)\right] \Delta Q^n = -\Delta\tau R^n$$

where (7)

$$R^n = \delta_i F^n + \delta_j G^n + \delta_k H^n - \delta_j G_v^n - \delta_k H_v^n$$

The inviscid flux Jacobians A, B, and C and viscous flux Jacobians $B_v$ and $C_v$ terms arising from linearization are given by

$$A = \frac{\partial F}{\partial Q}\ ,\ B = \frac{\partial G}{\partial Q}\ ,\ C = \frac{\partial H}{\partial Q}\ ,\ B_v = \frac{\partial G_v}{\partial Q}\ ,\ C_v = \frac{\partial H_v}{\partial Q}$$

## 2. BOUNDARY CONDITIONS

It is well known that, when dealing with a time–marching formulation, the reflecting behavior of the numerical boundary conditions must be minimized in order to enhance the convergence rate to the steady state. In the present work all boundary conditions are explicitly implemented. They include farfield, solid wall, and block–to–block boundary conditions. All farfield (i.e. inflow–outflow) and inviscid impermeable boundaries used characteristic variable boundary conditions as derived in Reference [4] for stationary grids and in Reference [5] for dynamic grids. As in these references, the boundary conditions are implemented utilizing one layer of points outside of the computational field, called phantom points, which results in first–order accuracy at the boundaries. The change in dependent variable, $\Delta Q^n$, is set to zero for all boundaries except at block–to–block boundaries. The approach adopted here with regard to block–to–block interface communication is a direct extraction–injection procedure which was taken by Belk [Reference 5] . This means the information (such as Q, $\Delta Q$) from within the domain of one block can be extracted and then injected as phantom data in an adjacent block. In Reference [5], Belk investigated many of the dilemmas posed when attempting time–accurate flowfield simulations using dynamic blocked grids. Belk showed that using synchronized dependent variables and approximating the value of the solution vector required at block boundaries with whatever information is currently available from adjoining blocks introduces an $O((\Delta t)^2/\Delta x)$ error at the boundary and gives unsteady results that compare well with unblocked results even for cases with a shock wave passing through the block boundary. For viscous wall boundaries, a no–slip implementation of the zero normal pressure gradient boundary condition is applied at the body surface. In the case of wall heat transfer, the wall temperature $T_w$ is specified, and the density at

the wall is computed from the equation of state, knowing the surface pressure and surface temperature, to wit

$$
\begin{aligned}
p_p &= p_f = p_w \\
\vec{V}_p &= 2\vec{X}_w - \vec{V}_f \\
\varrho_w &= (\gamma p_w)/T_w \\
\varrho_p &= 2\varrho_w - \varrho_f \\
e_p &= (\gamma - 1)^{-1} p_p + \frac{1}{2} \varrho_p (u_p^2 + v_p^2 + w_p^{2)}
\end{aligned}
$$

where $\vec{X}_w$ is wall grid speed and the subscripts p, f, and w denote phantom points, field points, and wall boundary points respectively.

## 3. TURBULENCE MODEL

Closure of the governing equations is achieved by using a turbulence model to obtain the eddy viscosity values. In this study, the viscosity coefficient in Equation (2) for turbulent flow is modeled as the sum of the laminar and turbulent viscosity in the eddy viscosity approach. The turbulent eddy viscosity $\mu_t$ is computed by using the algebraic viscosity model due to Baldwin and Lomax [Reference 6]. It is a two–layer model in which an eddy viscosity is calculated for an inner and outer region. The inner region follows the Prandtl–van Driest formulation. In both the inner and outer formulations the distribution of vorticity is used to determine the length scales, thereby avoiding the necessity of finding the outer edge of boundary layer. For the inner region,

$$\mu_{t_{inner}} = \varrho \ell^2 |\omega|$$

where

$$\ell = ky\left[1 - \exp\left(\frac{-y^+}{26}\right)\right]$$

and

$$y^+ = \frac{y}{\mu_w}(\varrho_w \tau_w)^{\frac{1}{2}}$$

y is the normal distance from the wall, $|\omega|$ is the absolute magnitude of vorticity, and k=0.4 is the von Karman constant.

The eddy viscosity for the outer region is given by

$$\mu_{t_{outer}} = KC_{cp}F_{Wake}F_{Kleb}(y)$$

where

$$F_{wake} = \text{minimum}\left( y_{max} F_{max} , C_{wk} y_{max} \frac{U_{diff}^2}{F_{max}} \right)$$

The quantities $y_{max}$ and $F_{max}$ are determined from the function

$$F(y) = y|\omega| \left[ 1 - \exp\left( \frac{-y^+}{26} \right) \right]$$

where $F_{max}$ is the maximum value of F(y) and $y_{max}$ is the value of y at which it occurs. The function $F_{kleb}(y)$ is the Klebanoff intermittency factor determined from

$$F_{kleb}(y) = \left[ 1 + 5.5 \left( \frac{C_{kleb}}{y_{max}} \right)^6 \right]^{-1}$$

The quantity $U_{diff}$ is the difference between the maximum and minimum total velocity in the profile and, for boundary layers, the minimum is defined as zero.

It is necessary to specify the following constants: $C_{cp}$=1.6, $C_{kleb}$=0.3, $C_{wk}$=0.25, and K=0.0168 is the Clauser constant.

## SECTION III

## NUMERICAL FLUX FORMULATION

This section is concerned with the numerical flux formulation of the convective terms. In this work the equations are discretized into a cell–center finite volume formulation that depends on the numerical flux at each cell face of the finite volume. There are, of course, many methods available now to determine this cell–face numerical flux. These methods range from central differencing to high resolution upwind schemes [Reference 7]. Simpson [Reference 8] demonstrates that exceptional results for a laminar viscous flow can be obtained using a high resolution method based on Roe's approximate Riemann solver [Reference 9] with only a few points in the viscous region compared to a flux vector split scheme [References 2 and 10] with many more points.

During the course of this research it became apparent that the approximate Riémann solver of Roe has much to offer with regard to the quality of the numerical solution of the Euler and Navier–Stokes equations. There are various methods in use for extending the first–order Roe scheme for the convective fluxes to higher order. Two of these methods will be discussed and a third method will be presented which was developed and used for all results presented in this report.

For multidimensional flow the assumption is made that the waves move normal to the cell face. Therefore, it is sufficient to present the numerical flux vector in only one dimension, as the same formulation is used in the other coordinate directions.

The first–order numerical flux, $f^*$, at cell face i + 1/2 resulting from Roe's approximate Riemann solver can be expressed by any one of the following three relations

$$f^*_{i+\frac{1}{2}} = [\, f(Q_i)\,]_{i+\frac{1}{2}} + \sum_{j=1}^{n} \alpha_{j,i+\frac{1}{2}}\ \lambda^{-(j)}_{i+\frac{1}{2}}\ r^{(j)}_{i+\frac{1}{2}} \tag{8}$$

$$f^*_{i+\frac{1}{2}} = [\, f(Q_{i+1})\,]_{i+\frac{1}{2}} - \sum_{j=1}^{n} \alpha_{j,i+\frac{1}{2}}\ \lambda^{+(j)}_{i+\frac{1}{2}}\ r^{(j)}_{i+\frac{1}{2}} \tag{9}$$

or

$$f^*_{i+\frac{1}{2}} = \frac{1}{2}[\, f(Q_i) + f(Q_{i+1})\,]_{i+\frac{1}{2}} - \frac{1}{2} \sum_{j=1}^{n} \alpha_{j,i+\frac{1}{2}} \left|\lambda^{(j)}_{i+\frac{1}{2}}\right| r^{(j)}_{i+\frac{1}{2}} \tag{10}$$

where

$$\left|\lambda^{(j)}_{i+\frac{1}{2}}\right| = \lambda^{+(j)}_{i+\frac{1}{2}} - \lambda^{-(j)}_{i+\frac{1}{2}} \tag{11}$$

$$\alpha_{j,i+\frac{1}{2}} = \ell^{(j)}_{i+\frac{1}{2}} \cdot (Q_{i+1} - Q_i) \tag{12}$$

and n = 5 for three–dimensional fluid flow. In Eqs.(8–10), $\lambda^{\pm(j)}$ correspond to the positive and negative eigenvalues of the Roe matrix, $r^{(j)}$ are the right eigenvectors of the Roe matrix, $\ell^{(j)}$ are left eigenvectors of the Roe matrix, and the scalars $\alpha_j$ are jumps in the characteristic variables. The subscript i+1/2 in the equations above indicates that the metrics used correspond to the cell face located at i+1/2. The dependent variables in the eigenvalues and eigenvectors are, or course, the Roe averaged variables [Reference 9] at cell face i+1/2 since they comprise the eigensystem of the Roe matrix. The flux vectors f ($Q_i$) and f($Q_{i+1}$) in Equations (8–10) are evaluated using the dependent variable vector as indicated (not the Roe averaged variables), but the subscript i+1/2 on the bracket indicates the metrics at i+1/2 are to be used.

Equations (8–10) can also be written, respectively, as

$$f^{*}_{i+\frac{1}{2}} = [\, f(Q_i) \,]_{i+\frac{1}{2}} + \overline{A}^{-}(Q_i, Q_{i+1})\ (Q_{i+1} - Q_i) \tag{13}$$

$$f^{*}_{i+\frac{1}{2}} = [\, f(Q_{i+1}) \,]_{i+\frac{1}{2}} - \overline{A}^{+}(Q_i, Q_{i+1})\ (Q_{i+1} - Q_i) \tag{14}$$

$$f^{*}_{i+\frac{1}{2}} = \frac{1}{2}[\, f(Q_i) + f(Q_{i+1}) \,]_{i+\frac{1}{2}} - \frac{1}{2}|\overline{A}(Q_i, Q_{i+1})|\ (Q_{i+1} - Q_i) \tag{15}$$

where $\overline{A}$ is the usual diagonalized Roe matrix composed of Roe averaged variables with

$$\overline{A}^{\pm} = T\Lambda^{\pm}T^{-1} \tag{16}$$

$$|\overline{A}| = \overline{A}^{+} - \overline{A}^{-} \tag{17}$$

The matrix T in Equation (16) has as its columns the right eigenvectors, $r^{(j)}$, of the Roe matrix. The matrices $\Lambda^{\pm}$ are diagonal matrices with eigenvalues, $\lambda^{\pm(j)}$, of the Roe matrix along the diagonal.

One of the more straightforward and frequently used methods for extending the Roe scheme to higher order is the so–called MUSCL approach introduced by Van Leer [Reference 11]. This approach is a dependent variable extrapolation method whereby a dependent variable on either side of a cell face, denoted $Q_R$ and $Q_L$, is extrapolated up to either side of the cell face and these extrapolated variables are then used in the Roe formulation, e.g. Equation (15), to compute the numerical flux at a cell face. The higher–order Roe flux could then be computed from the relation

$$f^*_{i+1/2} = \frac{1}{2}[f(Q_L) + f(Q_R)]_{i+1/2} - \frac{1}{2}|\overline{A}(Q_L, Q_R)|\,(Q_R - Q_L) \tag{18}$$

where $Q_{i+1}$ has been replaced by $Q_R$, and $Q_i$ has been replaced by $Q_L$.

It should also be noted that limiting can be used in this formulation by including limiters in the computation of the extrapolated dependent variables $Q_R$ and $Q_L$ as was done by Anderson, Thomas, and Van Leer [Reference 12]. Extrapolation formulas without limiting are also presented in Reference 12 as well as Reference 7.

Another approach is the numerical flux family of higher-order accurate TVD schemes using Roe averaging [Reference 9] introduced by Osher and Chakravarthy [Reference 13]. The numerical flux for this family of TVD schemes for up to third-order can be written as [Reference 14]

$$\begin{aligned} f^*_{i+1/2} = {} & \frac{1}{2}[f(Q_i) + f(Q_{i+1})]_{i+1/2} - \frac{1}{2}\sum_{j=1}^{n} (\sigma^+_{j,i+1/2} - \sigma^-_{j,i+1/2})\, r^{(j)}_{i+1/2} \\ & + \sum_{j=1}^{n}\left[\frac{1-\psi}{4} L_j^+(-1,1)\, r^{(j)}_{i-1/2} + \frac{1+\psi}{4} L_j^+(1,-1)\, r^{(j)}_{i+1/2}\right] \\ & - \sum_{j=1}^{n}\left[\frac{1-\psi}{4} L_j^-(3,1)\, r^{(j)}_{i+3/2} + \frac{1+\psi}{4} L_j^-(1,3)\, r^{(j)}_{i+1/2}\right] \end{aligned} \tag{19}$$

where

$$\sigma^{\pm}_{j,i+1/2} = \lambda^{\pm(j)}_{i+1/2}\, \alpha_{j,i+1/2} \tag{20}$$

and again n = 5 for three-dimensional fluid flow. The parameter $\psi$ is typically -1,0, or 1/3 and controls the magnitude of the truncation error without limiting [Reference 15].

The functions $L_j^{\pm}(\ell, m)$ used in Equation (19) are limiters. Three limiters were used; the minmod, Superbee, and Van Leer. The minmod limiter is:

$$L_j^{\pm}(\ell, m) = \text{minmod}\left(\sigma^{\pm}_{j,i+\ell/2},\ b\sigma^{\pm}_{j,i+m/2}\right) \tag{21}$$

where

$$\text{minmod}(x, y) = \text{sign}(x)\ \max\{0,\ \min[|x|,\ y\ \text{sign}(x)]\} \tag{22}$$

$$b = \frac{3-\psi}{1-\psi} \tag{23}$$

and the parameter b is a compression parameter [References 13, 16]. The Superbee limiter is due to Roe and is:

$$L_j^{\pm}(\ell, m) = \text{cmplim}\left(\sigma^{\pm}_{j,i+\ell/2}, \sigma^{\pm}_{j,i+m/2}\right) \tag{24}$$

where

$$\text{cmplim}(x, y) = \text{sign}(x)\ \max\{0,\ \min[|x|,\ \beta\ y\ \text{sign}(x)], \min[\beta\ |x|, y\ \text{sign}(x)]\} \qquad (25)$$

The compression parameter $\beta$ in Equation (25) is stated by Sweby [Reference 17] to be in the range $1 \leq \beta \leq 2$ ($\beta = 2$ is typically used). The Van Leer limiter is:

$$L_j^{\pm}\ (\ell,\ m) = \text{vl}\left(\sigma^{\pm}_{j,i+\ell/2}, \sigma^{\pm}_{j,i+m/2}\right) \qquad (26)$$

where

$$\text{vl}\ (x, y) = \frac{xy + |xy|}{x + y} \qquad (27)$$

Note in Equation (19) that although the numerical flux at cell face i+1/2 is being computed, metrics and Roe variables at cell faces i–1/2 and i+3/2 are involved. The additional terms added to the first–order numerical flux in Equation (19) to achieve higher–order accuracy might be viewed as an extrapolation of flux differences, as opposed to the extrapolation of dependent variables as in the MUSCL approach.

A third approach to obtaining a higher–order numerical flux, and the one used in this work, is one that falls somewhere between the dependent variable extrapolation (MUSCL) and the flux extrapolation approaches. Since any of Equations (8), (9), or (10) give the same first–order numerical flux, and because the evaluation of Equation (10) requires a slightly larger number of floating point operations than either of Equations (8) or (9), Equation (8) is used for the first–order numerical flux along with certain additional terms to obtain the following higher–order numerical flux

$$f^{*}_{i+1/2} = [f\ (Q_i)]_{i+1/2} + \sum_{j=1}^{n} \sigma^{-}_{j,i+1/2}\ r^{(j)}_{i+1/2} + \sum_{j=1}^{n}\left\{\frac{1-\psi}{4}[L_j^{+}(-1, 1) - L_j^{-}(3, 1)] + \frac{1+\psi}{4}\ [L_j^{+}(1, -1) - L_j^{-}(1, 3)]\right\} r^{(j)}_{i+1/2} \qquad (28)$$

where the $\sigma^{\pm}$ used in the limiters are now defined as

$$\sigma^{\pm}_{j,i+P/2} = \lambda^{\pm\,(j)}_{i+1/2}\ \alpha_{j,i+P/2} \qquad (29)$$

where

$$\sigma_{j,i-1/2} = l^{(j)}_{i+1/2} \cdot (Q_i - Q_{i-1}) \tag{30a}$$

$$\sigma_{j,i+1/2} = l^{(j)}_{i+1/2} \cdot (Q_{i+1} - Q_i) \tag{30b}$$

$$\sigma_{j,i+3/2} = l^{(j)}_{i+1/2} \cdot (Q_{i+2} - Q_{i+1}) \tag{30c}$$

The limiters retain the same definition. However, when using Equations (29) and (30) in place of Equation (20), the Superbee and Van Leer limiters reduce to

$$L_j^{\pm}(\ell, m) = L_j^{\pm}(m, \ell) \tag{31}$$

and the scheme is independent of $\psi$ when these limiters are used. This would then appear to be like a second–order Fromm scheme which occurs for $\psi = 0$. Both second– and third–order accuracy can still be obtained with the minmod limiter.

Results obtained to date using Equation (28) have been superior to the results obtained using Equation (19). Another advantage of Equation (28) over Equation (19) is that the operation count is less. With regard to limiters, solutions have been obtained using each of the three limiters; minmod, Superbee, and Van Leer. The solutions differ, but for many problems the solutions are reasonably close. For the most part, Van Leer and second– and third–order minmod give about the same results. Superbee is a more compressive limiter as it was evidently designed to be. The biggest difference in the limiters has to do with convergence. Van Leer has always given the best convergence rate and Superbee usually gives the worst. Minmod has shown tendencies to go into limit cycles. Considering both quality of results and convergence rate, Van Leer is presently the suggested limiter. In view of the property of the Van Leer limiter expressed by Equation (31), the second–order numerical flux at a cell face can be written

$$\begin{aligned} f^*_{i+1/2} = [f(Q_i)]_{i+1/2} &+ \sum_{j=1}^{n} \sigma^-_{j,i+1/2}\, r^{(j)}_{i+1/2} \\ &+ \frac{1}{2} \sum_{j=1}^{n} [L_j^+(-1,1) - L_j^-(1,3)]\, r^{(j)}_{i+1/2} \end{aligned} \tag{32}$$

where $L_j^{\pm}(\ell, m)$ is given by Equation (26), or Equation (24) for that matter, since Superbee satisfies Equation (31) and $\sigma$ and $\alpha$ are given by Equations (29) and (30).

## SECTION IV

## NUMERICAL SOLUTION SCHEMES

In this cell–centered finite volume formulation the method used to obtain the numerical flux at a cell face has, of course, a strong influence on the quality of the numerical solution. The method used for the numerical flux at a cell face also has an influence on the choice of the numerical method used to solve the resulting system of algebraic equations. For example, the use of a flux vector split scheme [Reference 18] for obtaining the numerical flux at a cell face leads naturally to a solution matrix composed of elements corresponding to the Jacobian of the flux vector represented by only positive eigenvalues and the Jacobian of the flux vector represented by only negative eigenvalues. The solution matrix resulting from this flux vector split formulation leads more or less naturally to a factored LU type solution scheme [Reference 19] that can be coded into a rather efficient algorithm. This factored LU type scheme was put forth by Buning and Steger [Reference 20] for the solution of a two–dimensional finite difference formulation of the equations.

During the course of this research various implicit schemes were developed and used for the numerical solution of the three–dimensional unsteady Euler and Navier–Stokes equations. The purpose of this Section is to present these implicit methods and illustrate how each was developed in succession. This is carried out by first reviewing Newton's method for the solution of nonlinear systems of equations because this is the basic form in which the equations are cast in order to gain generality with regard to the solution of the unsteady equations so that the equations can be converged at each time step if desired.

### 1. NEWTON'S METHOD

Consider the system of nonlinear equations written in the general form

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0 \\ F_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ F_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{33}$$

These equations could, for example, be the three–dimensional unsteady Euler equations where $n=5$ and $F_1$ through $F_5$ are the equations for mass, momentum, and energy, and components $x_1$ through $x_5$ are the conserved variables of density, momentum, and energy. Equations (33) could also be a system of nonlinear algebraic equations resulting from the discretization of the three–dimensional unsteady Euler equations on a computational grid. The number of equations and

dependent variables can then be extremely large due to the fact that there are five equations and five unknown dependent variables for every cell or node point in the computational domain. Of particular interest here is the latter case, where a solution is required for the large number of nonlinear algebraic equations that result from a finite volume discretization of the unsteady three–dimensional Euler or Navier–Stokes equations.

Let F be a vector function of the coordinate functions $F_1, F_2, ...., F_n$ where $x_1, x_2, ..., x_n$ are coordinates of the vector x. Then the system of equations given by Equations (33) can be written as

$$F(x) = 0 \tag{34}$$

Newton's method for the vector function F(x) can be written (see, for example, Ortega and Rheinboldt [Reference 21]) as

$$x^{m+1} = x^m - (F'(x^m))^{-1} F(x^m) \tag{35}$$

where m = 1,2,3,... and F'(x) is the Jacobian matrix of the vector F(x) and is given by

$$F'(x) = \begin{bmatrix} a_{11}(x) & a_{12}(x) & \dots & a_{1n}(x) \\ a_{21}(x) & a_{22}(x) & \dots & a_{2n}(x) \\ \vdots & \vdots & & \vdots \\ a_{n1}(x) & a_{n2}(x) & \dots & a_{nn}(x) \end{bmatrix} \tag{36}$$

where

$$a_{ij}(x) = \frac{\partial F_i(x)}{\partial x_j} \tag{37}$$

It is usually impractical to obtain the inverse of the matrix operator F'(x). A more practical way of writing Newton's method for obtaining numerical solutions is

$$F'(x^m)(x^{m+1} - x^m) = -F(x^m) \tag{38}$$

Two good references on Newton's method for nonlinear systems of equations are the books by Ortega and Rheinboldt [Reference 21] and Dennis and Schnabel [Reference 22]. By taking advantage of these (and other) works on Newton's method, it is a rather simple matter to state the problem. However, it is an entirely different matter to numerically solve the resulting system of equations.

## 2. NEWTON'S METHOD COMPARED TO NORMAL LINEARIZATION

It is interesting to compare the equations resulting from the application of Newton's method with the equations resulting from what is referred to here as normal linearization as introduced in References [23 and 24]. For illustration purposes, consider the first–order nonlinear hyperbolic system that has been transformed from x, t space to $\xi$, $\tau$ space

$$\frac{\partial Q}{\partial \tau} + \frac{\partial f(Q)}{\partial \xi} = 0 \tag{39}$$

Discretizing this equation in the finite volume sense with $\Delta\xi = 1$, and using a first–order backward difference in time, the resulting system of difference equations that must be solved can be written as

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta\tau} + f_{i+1/2}(Q^{n+1}) - f_{i-1/2}(Q^{n+1}) = 0 \tag{40}$$

or

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta\tau} + \delta_i f(Q^{n+1}) = 0 \tag{41}$$

The normal method introduced in References [23 and 24] to linearize Equation (41) would be to linearize the vector function f(Q) to obtain

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta\tau} + \delta_i (f(Q^n) + f'(Q^n)(Q^{n+1} - Q^n)) = 0 \tag{42}$$

where $f'(Q)$ is the Jacobian matrix of the vector function $f(Q)$. In the CFD literature Equation (42) is usually written

$$(I + \Delta\tau\, \delta_i A \cdot)(Q^{n+1} - Q^n) = -\Delta\tau\, \delta_i f(Q^n) \tag{43}$$

where

$$A = f'(Q^n) \tag{44}$$

On the other hand, to solve Equation (41) using Newton's method, the left hand side of Equation (41) is simply the vector function F given by Equation (34) and the vector x in Equation (34) is the dependent variable vector Q at time level n+1, i.e. $x = Q^{n+1}$. Newton's method is now formulated, and the system of difference equations that must be solved is given by Equation (38); or, in the nomenclature of this section, the equations that must be solved are given by

$$(I + \Delta\tau\, \delta_i\, A\, \cdot)\,(Q^{n+1,m+1} - Q^{n+1,m}) = -\Delta\tau \left[ \frac{Q_i^{n+1,m} - Q_i^n}{\Delta\tau} + \delta_i\, f\,(Q^{n+1,m}) \right] \tag{45}$$

where in this case

$$A = f\,'(Q^{n+1,m}) \tag{46}$$

Note that if the initial approximation (m=1) to $Q^{n+1}$ of Equation (45) is taken as

$$Q^{n+1,1} \rightarrow Q^n \tag{47}$$

then the first iteration of Equation (45) is the same as the solution $Q^{n+1}$ of Equation (43). An obvious difference between the normal linearization resulting in Equation (43), and Newton's method resulting in Equation (45), is that the time derivative is contained in the residual term (right hand side of the equation) in Equation (45), whereas, the residual term in Equation (43) does not contain the time derivative. For steady state problems where time could be an iteration parameter, continued iteration of Equation (43) would presumably lead to

$$Q^{n+1} \rightarrow Q^n \tag{48}$$

and

$$\delta_i\, f\,(Q^n) \rightarrow 0 \tag{49}$$

On the other hand, for unsteady problems, a solution of Equation (43) does not insure that

$$\frac{Q_i^{n+1} - Q_i^n}{\Delta\tau} + \delta_i\, f(Q^n) = 0 \tag{50}$$

whereas, a converged solution of Equation (45) does insure that the unsteady equation, Equation (41), is satisfied.

## 3. SOLUTION SCHEMES

Of particular interest here is the numerical solution of the so–called upwind formulation of the equations for three–dimensional time–dependent problems. An implicit three–factor scheme of the three–dimensional upwind equations requiring three block tridiagonal solutions similar to the scheme of References [23 and 24] was shown by Anderson [Reference 25] to be conditionally stable with a maximum CFL number of order 10. Anderson [Reference 25] also investigated the stability properties of a two–factor LU scheme considered in Reference 19, and found it to have improved stability properties compared to the three–factor scheme. This was also the conclusion of a more detailed analysis performed by Mansfield [Reference 26]. The two–factor

scheme (usually referred to as the two–pass scheme) is a workhorse scheme used to solve numerous steady and unsteady flow problems about rather complex three–dimensional configurations. Although the two–pass scheme has been described several times, it will be reviewed briefly again, because a new scheme will be presented that involves modifications of the two–pass scheme.

a. TWO–PASS SCHEME

Consider the three–dimensional extension of Equation (39) written as

$$\frac{\partial Q}{\partial \tau} + \frac{\partial f(Q)}{\partial \xi} + \frac{\partial g(Q)}{\partial \eta} + \frac{\partial h(Q)}{\partial \zeta} = 0 \tag{51}$$

The finite volume discretization of Equation (51) analogous to Equation (41) is

$$\frac{Q^{n+1}_{i,j,k} - Q^{n}_{i,j,k}}{\Delta \tau} + \delta_i f(Q^{n+1}) + \delta_j g(Q^{n+1}) + \delta_k h(Q^{n+1}) = 0 \tag{52}$$

The flux vector split form of the equations (see, for example, Reference 19) can be written as

$$\frac{Q^{n+1}_{i,j,k} - Q^{n}_{i,j,k}}{\Delta \tau} + \delta_i[f^+(Q^{n+1}) + f^-(Q^{n+1})] + \delta_j[g^+(Q^{n+1}) + g^-(Q^{n+1})] + \delta_k[h^+(Q^{n+1}) + h^-(Q^{n+1})] = 0 \tag{53}$$

Newton's method for this three spatial dimension problem that is analogous to Equation (45) is

$$[I + \Delta\tau(\delta_i A^+_{\cdot} + \delta_i A^-_{\cdot} + \delta_j B^+_{\cdot} + \delta_j B^-_{\cdot} + \delta_k C^+ + \delta_k C^-)]$$
$$(Q^{n+1,m+1} - Q^{n+1,m}) = -\Delta\tau R^{n+1,m} \tag{54}$$

where

$$A^{\pm} = \frac{\partial f^{\pm}(Q^{n+1,m})}{\partial Q^{n+1,m}}$$

$$B^{\pm} = \frac{\partial g^{\pm}(Q^{n+1,m})}{\partial Q^{n+1,m}}$$

$$C^{\pm} = \frac{\partial h^{\pm}(Q^{n+1,m})}{\partial Q^{n+1,m}}$$

$$R^{n+1,m} = \frac{Q^{n+1,m}_{i,j,k} - Q^{n}_{i,j,k}}{\Delta \tau} + \delta_i[f^+(Q^{n+1,m}) + f^-(Q^{n+1,m})] + \delta_j[g^+(Q^{n+1,m}) + g^-(Q^{n+1,m})] + \delta_k[h^+(Q^{n+1,m}) + h^-(Q^{n+1,m})]$$

Defining

$$\begin{aligned} X^m &= Q^{n+1,m+1} - Q^{n+1,m} \\ \delta M^+ &= \delta_i A^+ + \delta_j B^+ + \delta_k C^+ \\ \delta M^- &= \delta_i A^- + \delta_j B^- + \delta_k C^- \end{aligned} \tag{55}$$

Equation (54) can be written

$$(I + \Delta\tau\delta M^+ + \Delta\tau\delta M^-)X^m = -\Delta\tau R^{n+1,m} \tag{56}$$

The two–pass (LU) scheme for solving this system is

$$(I + \Delta\tau\delta M^+)(I + \Delta\tau\delta M^-)X^m = -\Delta\tau R^{n+1,m} \tag{57}$$

which can be solved in the following steps

$$\begin{aligned} (I + \Delta\tau\delta M^+)X^{1,m} &= -\Delta\tau\, R^{n+1,m} \\ (I + \Delta\tau\delta M^-)X^{2,m} &= X^{1,m} \\ Q^{n+1,m+1} - Q^{n+1,m} &= X^{2,m} \end{aligned} \tag{58}$$

b. MODIFIED TWO–PASS SCHEME

To obtain the modified two–pass scheme, first expand Equation (56) rather than factor it to obtain

$$X_i^m + \Delta\tau(M_i^+ X_i^m - M_{i-1}^+ X_{i-1}^m) + \Delta\tau(M_{i+1}^- X_{i+1}^m - M_i^- X_i^m) = -\Delta\tau R_i^{n+1,m} \tag{59}$$

(The nomenclature gets a bit sticky. The single subscript i represents the point in three–dimensional computational space corresponding to the point i,j,k. Subscript i+1 corresponds to i+1,j,k; i,j+1,k; or i,j,k+1. One simple way of following the argument is to think of this as a one–dimensional problem. Remember, however, that the matrix M is the sum of the flux Jacobians in each of the three computational directions, and subscript i+1 can represent any of the three adjoining cells that are in the positive computational direction of the cell i,j,k, and the subscript i–1 can represent any of the three adjoining cells that are in the negative computational direction of the cell i,j,k.) Dividing Equation (59) by the time step, $\Delta\tau$, one obtains

$$\left(\frac{1}{\Delta\tau}I + M^+ - M^-\right)_i X_i^m - M_{i-1}^+ X_{i-1}^m + M_{i+1}^- X_{i+1}^m = -R_i^{n+1,m} \tag{60}$$

or

$$D_i X_i^m - M_{i-1}^+ X_{i-1}^m + M_{i+1}^- X_{i+1}^m = - R_i^{n+1,m} \tag{61}$$

where

$$D = \frac{1}{\Delta\tau} I + M^+ - M^- \tag{62}$$

There are at least two ways of looking at the modified two–pass scheme. One is to view the method as a factored scheme, and another is to view the method as a relaxation scheme. Both points of view will be considered next.

(1) MODIFIED TWO–PASS AS A FACTORED SCHEME

Assuming D to be nonsingular, Equation (61) can be written as

$$X_i^m - D_i^{-1} M_{i-1}^+ X_{i-1}^m + D_i^{-1} M_{i+1}^- X_{i+1}^m = - D_i^{-1} R_i^{n+1,m} \tag{63}$$

Equation (63) can be factored just as the two–pass (LU) scheme to obtain

$$(I - D_i^{-1} M_{i-1}^+)(I + D_i^{-1} M_{i+1}^-) X^m = - D_i^{-1} R_i^{n+1,m} \tag{64}$$

Equation (64) can also be solved in two passes (two steps) by

$$(D_i - M_{i-1}^+) X^{1,m} = - R_i^{n+1,m}$$
$$(D_i + M_{i+1}^-) X^{2,m} = D_i X^{1,m}$$

or

$$\begin{aligned} D_i X^{3,m} + M_{i+1}^- X^{2,m} &= 0 \\ Q^{n+1,m+1} - Q^{n+1,m} &= X^{2,m} = X^{3,m} + X^{1,m} \end{aligned} \tag{65}$$

(2) MODIFIED TWO–PASS AS A RELAXATION SCHEME

Consider solving Equation (61) using the Gauss–Seidel method. In Equation (61) D is a block diagonal matrix, $M^+$ is a block lower triangular matrix with zeros on the diagonal, and $M^-$ is a block upper triangular matrix with zeros on the diagonal. Considering $X^{1,m}$ to be initially zero, the Gauss–Seidel method is

$$D_i X_i^{1,m} - M_{i-1}^+ X_{i-1}^{1,m} = - R_i^{n+1,m} \tag{66}$$

After completing this forward pass the $X^{1,m}$ are known, and a backward pass can then be carried out to complete a symmetric Gauss–Seidel solution by

$$D_i X_i^{2,m} - M_{i-1}^+ X_{i-1}^{1,m} + M_{i+1}^- X_{i+1}^{2,m} = - R_i^{n+1,m} \tag{67}$$

Some computational conveniences can be realized by noting the following. Using Equation (66) for the vector $M_{i-1}^{+}X_{i-1}^{1,m}$, one can write Equation (67) as

$$D_i X_i^{2,m} + M_{i+1}^{-} X_{i+1}^{2,m} = D_i X_i^{1,m} \tag{68}$$

so that in the backward pass one can actually solve for $X^{3,m}$, where

$$X^{3,m} = X^{2,m} - X^{1,m} \tag{69}$$

just as in Equations (65). This reduces the number of matrix–vector multiplications that would otherwise need to be carried out. The symmetric Gauss–Seidel relaxation scheme is then given by Equations (66), (68), and (69).

The end result of this is that both points of view (that is, whether the modified two–pass scheme is considered a factored scheme or a symmetric Gauss–Seidel relaxation scheme without residual updating) are the same.

It was recently discovered that the Russian Samarskii [Reference 27] put forth a scheme in 1964 that is similar to this modified two–pass scheme, and it is referred to in Reference [27] as the alternate–triangular method. The analysis of the alternate–triangular method presented in Reference [27] requires rather strict properties of the solution operators, and the present solution operators do not satisfy these properties. Application and analysis of the alternate–triangular method to elliptic difference equations is presented in Chapter 10 of the Russian book by Samarskii and Nikolaev [Reference 27].

### c. N–PASS SCHEME

It was shown above that the Modified Two–Pass Scheme could be viewed as a forward and backward pass of a symmetric Gauss–Seidel scheme if the initial guess for $X^{1,m}$ was taken as zero and the solution process was terminated after one forward and one backward pass. By assuming an initial guess, denoted $X^{0,m}$, the first pass of this relaxation scheme analogous to Equation (66) can be written as

$$D_i X_i^{1,m} - M_{i-1}^{+} X_{i-1}^{1,m} + M_{i+1}^{-} X_{i+1}^{0,m} = - R_i^{n+1,m} \tag{70}$$

A backward pass analogous to Equation (67) can be written as

$$D_i X_i^{2,m} - M_{i-1}^{+} X_{i-1}^{1,m} + M_{i+1}^{-} X_{i+1}^{2,m} = - R_i^{n+1,m} \tag{71}$$

This process can, of course, be repeated, and is done so to obtain the N–Pass Scheme given by

$$D_i X_i^{p-1,m} - M_{i-1}^{+} X_{i-1}^{p-1,m} + M_{i+1}^{-} X_{i+1}^{p-2,m} = - R_i^{n+1,m} \tag{72}$$

and

$$D_i X_i^{p,m} - M_{i-1}^{+} X_{i-1}^{p-1,m} + M_{i+1}^{-} X_{i+1}^{p,m} = - R_i^{n+1,m} \tag{73}$$

where p = 2,4,6,...,2N.

From the indexing used above a better terminology for this method might be to call this a 2N–Pass Scheme, or perhaps, an N–Cycle Scheme if one forward and one backward pass through the computational domain were considered to constitute one cycle. Regardless of what constitutes a pass or a cycle, notice that this scheme, given by Equations (72) and (73), is simply symmetric Gauss–Seidel.

## 4. DISCRETIZED NEWTON METHOD

All terms, by conventional standard, appearing in the Equation (54) should result from a single flux formulation. Barth [Reference 28] encountered difficulty with the formal linearization of the Roe flux functions. He concluded that although superior convergence rate could be obtained using the formal linearization, the computational expense made it unattractive. Good results over the past few years have been obtained by a hybrid Roe scheme, which utilized the true positive and negative Jacobians derived from flux–vector splitting on the left–hand–side of Equation (54) and the residual term R on the right–hand–side of Equation (54) from flux–difference splitting. However, one could say that the hybrid Roe scheme is inconsistent in that the inviscid flux Jacobians used in the solution operator correspond to a flux–vector splitting scheme, while the residual vector is computed using the Roe scheme. A method which resolves this inconsistency when the Roe scheme is used to evaluate the inviscid flux vectors, is given in Reference[29]. This approach will work in principle for any scheme for which the true Jacobians are difficult to derive or approximate. In this procedure, the inviscid flux Jacobian matrices are computed numerically by discretization of the inviscid flux vector. Ortega and Rheinboldt [Reference 21] refer to such a method as a discretized Newton iteration. Various finite–difference approximations have been suggested in the numerical analysis literature. A simple and straightforward approximation of the elements of the Jacobian is to replace Equation (37) with

$$a_{ij}(x) = \frac{F_i(x + he_j) - F_i(x)}{h} \tag{74}$$

where $e_j$ is the jth unit vector. Dennis and Schnabel [Reference 22] point out that this is the same as approximating the jth column of the Jacobian by

$$\text{jth Column of } F'(x) = \frac{F(x + he_j) - F(x)}{h} \tag{75}$$

Dennis and Schnabel [Reference 22] also point out that if a sequence $\{h_m\}$ is used for the step size h, and if this sequence is properly chosen, then the quadratic convergence property of Newton's method is retained and Newton's method using finite differences is "virtually indistinguishable" from Newton's method using analytic derivatives. However, a straight Newton's method is not used here due to the magnitude of the problem in three dimensions, so since quadratic convergence is lost anyway it did not seem fruitful at this stage of research to invest significant operation count in sequencing h. In fact, a constant h of about half the reliable digits of the machine has worked as well thus far as using a variable step size. For example, on a Cray Y–MP in single precision (64 bits), constant step sizes of $h = 10^{-5}$ and $h = 10^{-8}$ gave virtually indistinguishable results. Therefore, for a constant step size, the present suggestion is

$$h = \sqrt{\text{machine epsilon}} \tag{76}$$

It should be noted that this approach of using a constant step size may not be appropriate for situations where there may be large variations in the magnitude of the elements of the dependent variable vector, although it has worked thus far for all cases considered. The application of the above mentioned procedure has been extended to evaluate the viscous flux Jacobian matrices as well. The viscous flux Jacobian matrices are computed numerically by discretization of the viscous flux vector.

## SECTION V

## SOFTWARE OPTIMIZATION

### 1. QUASI–BLOCK–TRIDIAGONAL SOLUTION MATRIX STRUCTURE

This subsection is dedicated to the analysis of the matrix form involved with the system $M\Delta Q = R$, i.e. $(Ax = b)$. Before discussing a couple of solution methods for solving this kind of a system, the general appearance of the matrices will be considered to see what can be done to simplify the problem. In this subsection, the structure of the matrices which result from solving the fluid dynamic equations in one, two, and three dimensions is presented. Do not be concerned with the actual entries in the matrices, as one can easily go back and see where each individual value came from. The goal is merely to get some pictures in mind to set the stage for taking advantage of particular characteristics of the system matrix in the solution methods which follow.

For the one–dimensional case, the form of the M matrix (referred to here as the system matrix) is very simple, and is shown in the following figure:

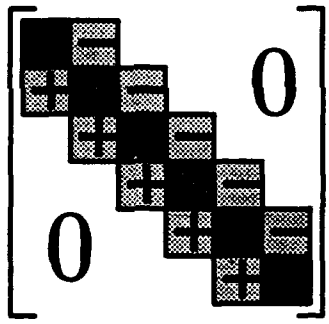



Figure 1. One–Dimensional Matrix Structure

In this case, all the entries (referred to here as point–blocks) are square and of order three. This is what is referred to as a "block–tridiagonal" matrix since there is a main diagonal of point–blocks, one super–diagonal, and one sub–diagonal, and the point–blocks which constitute them are all square and of the same size. Obviously, Gaussian reduction, a block version of Thomas' algorithm or approximate factorization are straight forward approaches to solving this matrix.

Extending from this one–dimensional case to see the two–dimensional system matrix structure, flux Jacobians in the second dimension are added to the matrix structure. In Figure 2, all point–blocks are of order four (conservation of mass, two components for conservation of momentum, and conservation of energy). The system is now banded with five bands (referred to here as block–pentadiagonal).

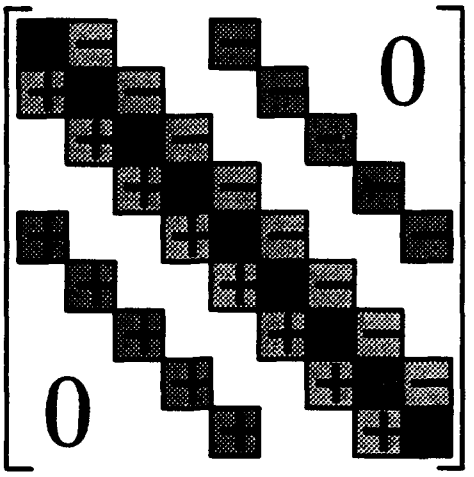


Figure 2. Two–Dimensional System Matrix Structure

Finally, for the three–dimensional case, there are now seven point–blocks of order five to place in the M matrix. Extending this system matrix structure of Figure 2 one dimension further, it can be shown that the matrix will be of the following form:

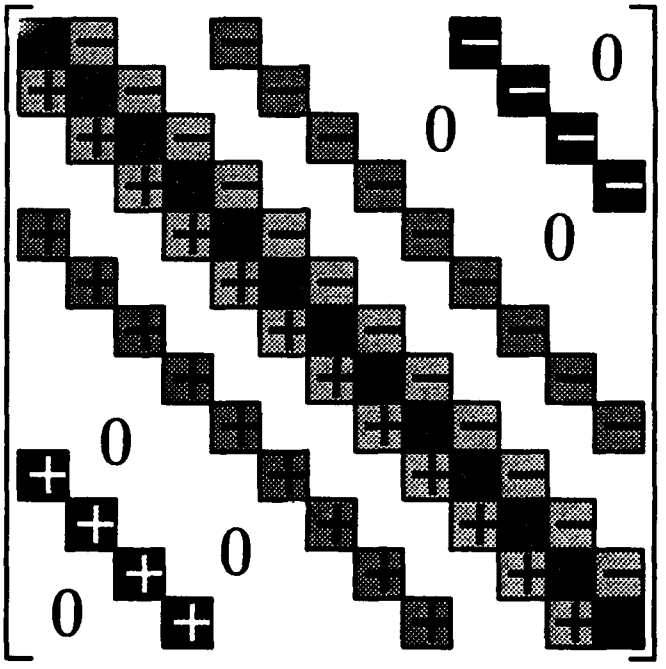


Figure 3. Three–Dimensional System Matrix Structure

In this case the system is block–septadiagonal, since there are three block diagonals both above and below the main block diagonal. Neither the 2D or 3D systems appear to have a very "clean" way of solving them, and the question of storage immediately comes to mind, since there are a tremendous number of zeros which must be kept. Even for a relatively small 2D geometry, it is not unusual to find a system with a few hundred block rows and columns. That im-

plies a few megawords of storage to hold the entire matrix, and perhaps less than 1% of that would (initially) be non-zero values.

The overriding question here is "How can solution concurrency be extracted from this kind of system without a monumental memory or solution convergence penalty?". Clearly, approximate factorization is attractive with regard to reducing the storage, but are there other methods which may take advantage of the banded nature of the system, without a huge cost penalty involved in storing all of the zeros? The research approach taken here involves analyzing the system, by observing the potential for making it "quasi-block-tridiagonal" as shall be discussed below.

Obviously, when dealing with the one-dimensional case, the system matrix is a straight forward block-tridiagonal system. Taking into account boundary conditions (explicit), the system matrix will take the form:

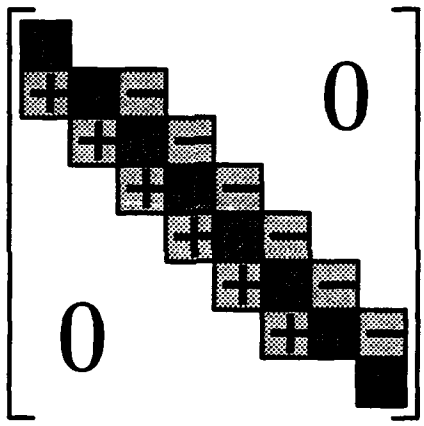



Figure 4. One-Dimensional System Matrix Structure with Boundary Conditions

The "I" point-blocks are identity matrices which simply set the boundary values to that determined explicitly and stored in the b vector. Of course, it is possible to have implicit boundary values thereby introducing additional point-blocks or altering the entries in point-blocks shown in this matrix. The simplest case (explicit boundary conditions) will be treated here since the principle concern will be how to find and utilize concurrency in these types of systems, and the boundary assumptions should not have an effect on these methods.

Extending to two-dimensional and three-dimensional systems, however, requires some work to get things looking like a "block-tridiagonal system". The simplest method is to take the M matrix with the boundary values included, and simply "draw lines" horizontally and vertically through the matrix, partitioning it into a peculiar looking block-tridiagonal structure. By doing this, groups of point-blocks (which are matrices) form what will be referred to here as matrix blocks.

Now a moment will be taken to look in some detail at what it really takes to partition a 3D problem from which the M matrix is built. First look at the very simple 3D space, which is shown below in Figure 5. The space is dimensioned as 5 x 4 x 4 ( i x j x k ). In addition, it has been partitioned into four planes running in the ij–plane. The reason for this is because of the type of software implementation which is typically used for assembling the M matrix.

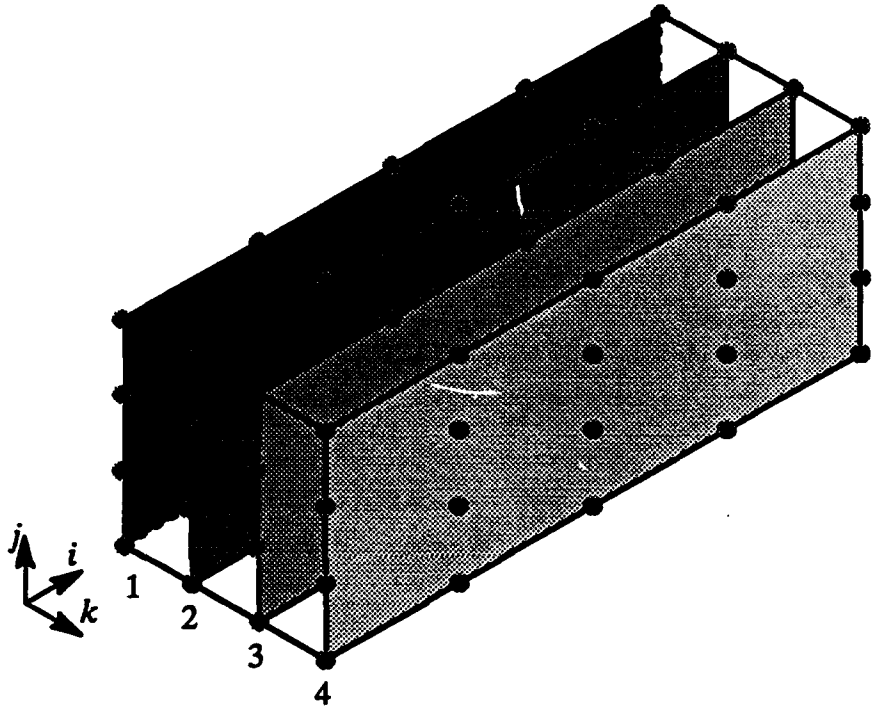



Figure 5. Three–Dimensional Computational Space

The dots on the surfaces (many of them are hidden behind the front–most plane) are the points (cells) where the fluid solution is sought. Traditionally, most 3D curvilinear spaces - associated with the solvers developed in association with this study are set up so that the max j and k indices are small in relation to the i index. It is common to establish the point distribution with a i >> j > k hierarchy in mind. An example, in "pseudo–code", of the way these points (equations) are assembled for processing is shown below.

```
for k=1 to 4
  for j=1 to 4
    for i=1 to 5
      Position this point's equation as the next row of the M matrix.
    end
  end
end
```

The M matrix which results from this type of arrangement is shown in Figure 6. This is the system which must be "block–tridiagonalized". In this case, the block–tridiagonal structure is found by simply drawing vertical and horizontal lines through the matrix as shown in the figure. The system has been partitioned, and the block–tridiagonal structure should be obvious. There are several other ways, obviously, to arrange the three loops, but all of them result in a similar block–tridiagonal arrangement.

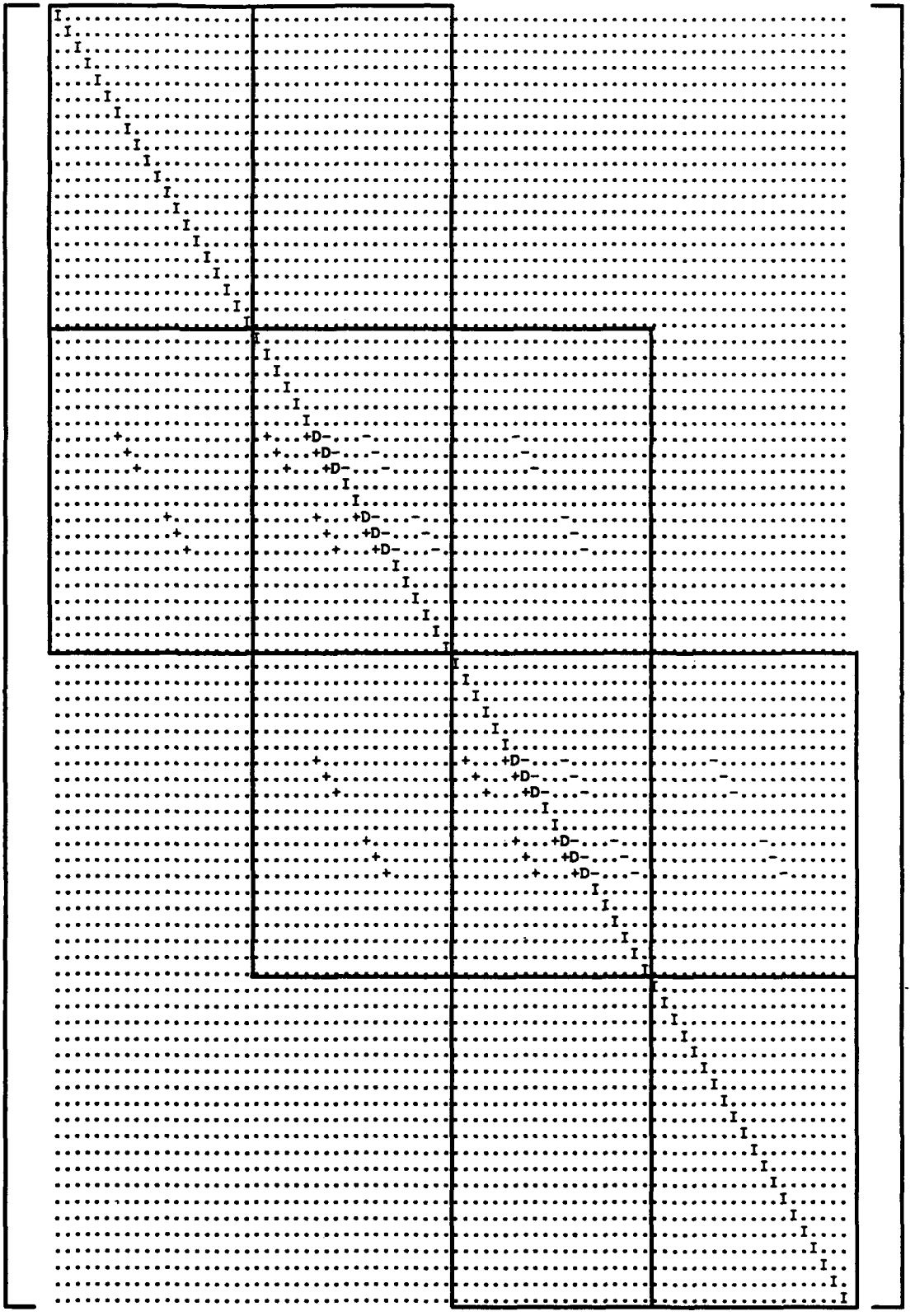


Figure 6. Typical Three–Dimensional System Matrix Structure (Tridiagonalized)

Next, concepts developed to extract natural concurrency finer than what has been viewed for traditional solution methods will be presented. The purpose of searching for fine–grained concurrency has been to provide opportunities for additional vectorization, such as what would be done on any Cray vector processor. The central theme here involves the use of the concept of "diagonal plane processing" for matrix ordering [References 5, 18].

The technique is to take a 2D or 3D space, and chose the ordering of points in the M matrix in such a way as to make the problem naturally appear as a quasi–block–tridiagonal system. This all boils down to traversing through a 3D space on a diagor ' plane in which none of the points in the plane are directly effecting any other. In the 2D case, this is analogous to traversing a diagonal line, and grouping the points falling on that line for concurrent processing. Here a brief look will be taken at each case.

In the 2D case, consider only the i and j directions. In Figure 7 below, consider a very small 2D space that has been partitioned into a 8 x 4 grid of interior points (cells). In addition, the boundary conditions (haloed points) are included, resulting in a 10 x 6 system, since the first and last rows and columns represent the boundary conditions. Also shown are every other diagonal line cutting the domain representing about half of the 15 cutting lines.

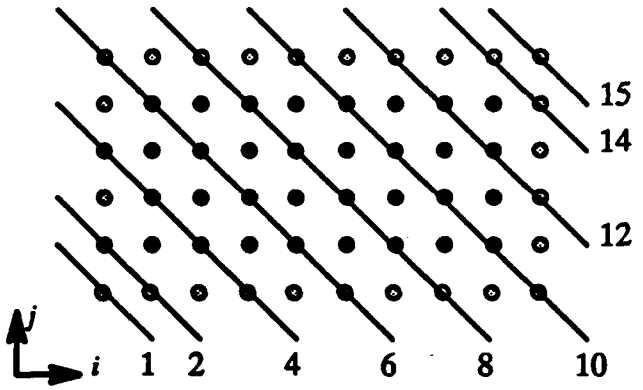



Figure 7. Two–Dimensional Computational Space with Diagonal Line Grouping

It can be shown, since all communication between points (from a typical difference stencil) is done in the i and j directions, no points lying on the same diagonal line are in direct communication. As a result of this, when looking at adjacent lines, the problem has reduced, in essence, to a one–dimensional problem. The result of this shows up in the M matrix organization, as shown in Figure 8. Again the identity point–blocks represent explicit boundary conditions. In this case, one takes the points from the top of each diagonal line, and works down, starting from line 1 and working through line 15. The blocks on the diagonal do not appear to be square. This is because the characters are taller than they are wide. However, the center blocks are all square, as is the overall matrix.

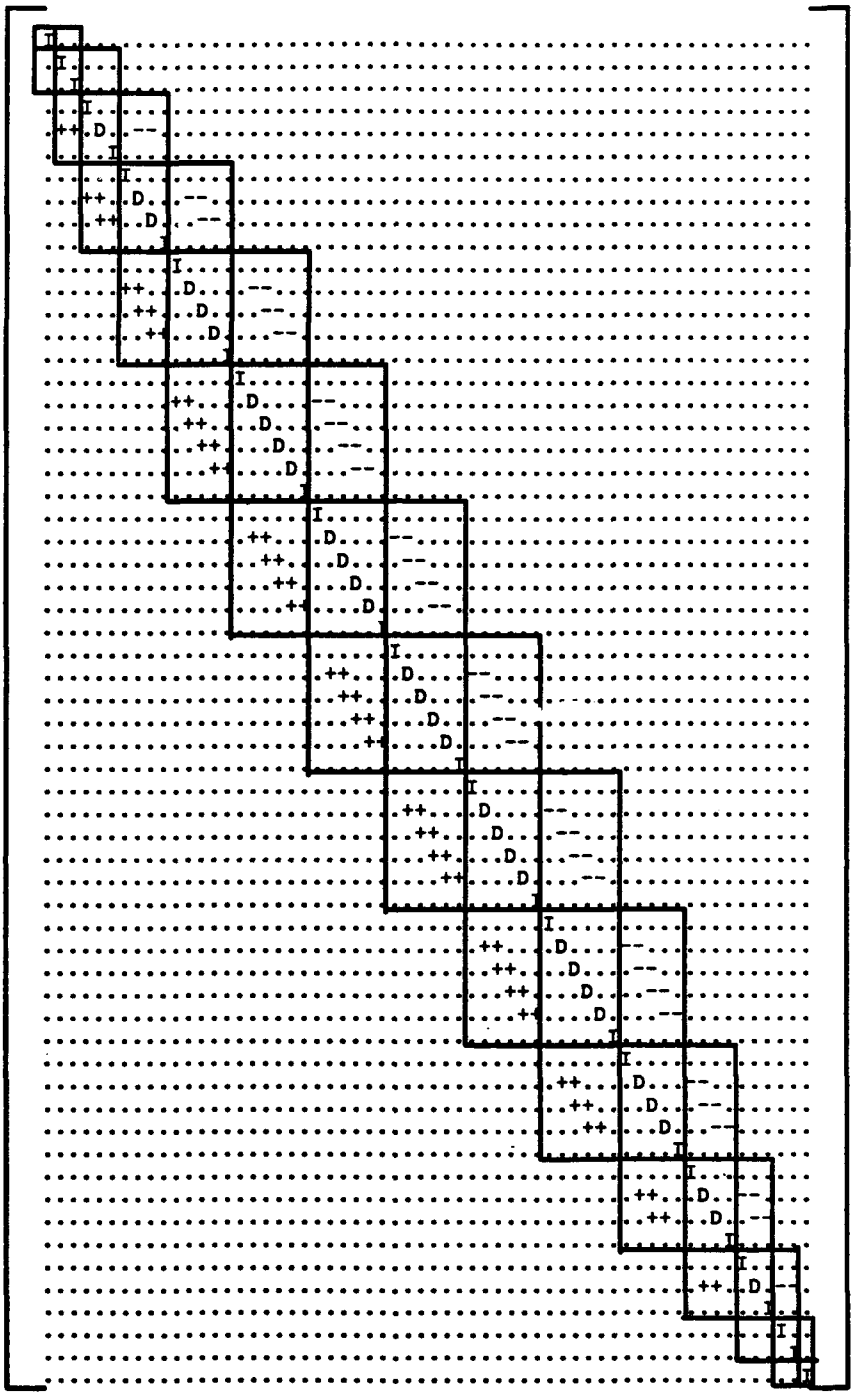


Figure 8. Two–Dimensional System Matrix Using Diagonal Line Ordering

This particular form is commonly referred to as "quasi–block–tridiagonal", or sometimes "block–tridiagonal with unequal matrix–block sizes". Systems which occur in these forms are common. In addition, the resulting banded system shown in Figure 8 has been "blocked" to show the block–tridiagonality in the system matrix.

Now consider the 3D case. The first thing needing to be done is to get a good picture in mind of what the 3D space looks like when partitioned into diagonal planes. The picture below, Figure 9, shows the 5 x 4 x 4 space ( i x j x k ) which has all of the diagonal planes shown. The origin of the space is found in the lower left hand corner, where the number "1" appears. Also, the diagonal plane intersections with the computational domain have been numbered from 1 through 11, with the first and last simply being single points in the corners.

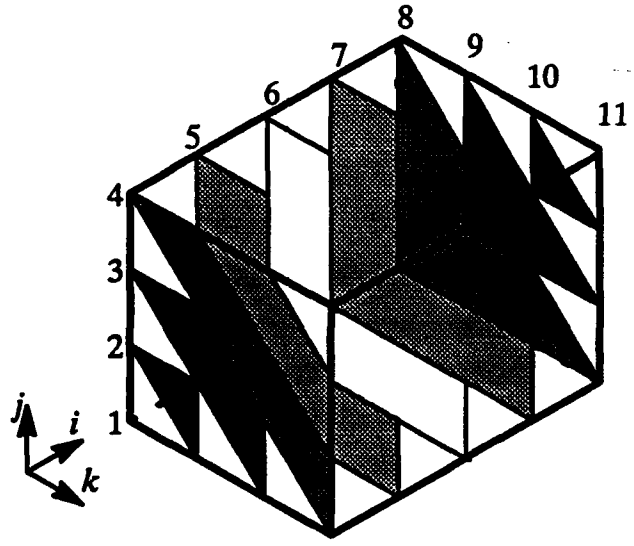



Figure 9. Diagonal Planes for 5 x 4 x 4 Computational Space

It may be a little helpful to look at each diagonal plane, so that the shape can be clearly seen, see Figure 10. Note how many points are contained on the perimeter and within each cutting plane. The points on the perimeter could represent the boundary points for example. By examining these points, the manner in which the boundary values fall into the M matrix can be seen.

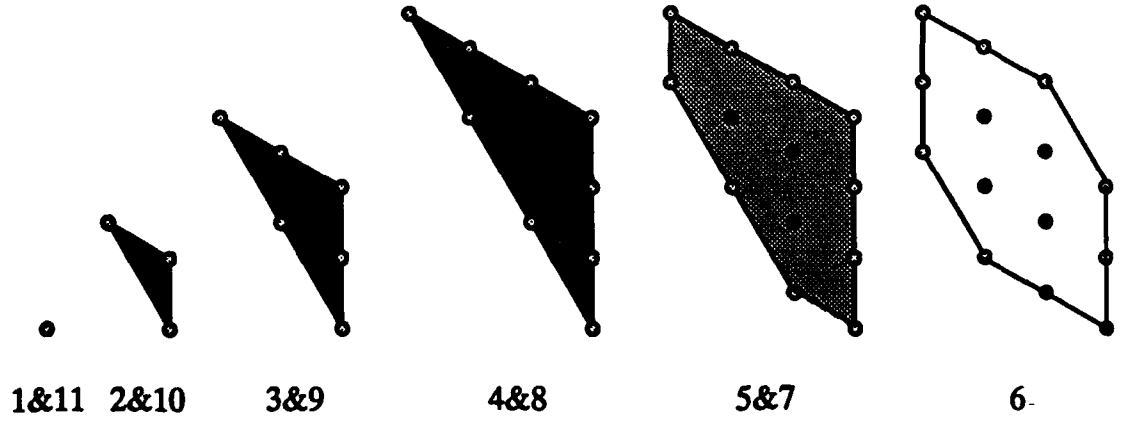



Figure 10. Individual Diagonal Planes

The resulting matrix form is only slightly more complex than in the 2D case. In this case, the points will be taken from consecutive planes 1 through 11. Within each plane the points will be ordered according to the following portion of "pseudo–code":

```
for SUM=3 to 14
  for k=1 to 4
    for j=1 to 4
      for i=1 to 5
          if i+j+k equals SUM then include the information
          (equation) for that point in the next row of the matrix.
      end
    end
  end
end
```

Since all of the points on a given diagonal plane have indices which add up to a constant (the plane number plus two), the above code will start with the first diagonal plane (which is a single point at the origin, with constant value of 1+1+1 = 3) and look for all points in the space which fall on each particular plane. That's the role of the first "for" loop. The next three merely control the order that the points are tested. Using the above algorithm for sampling the points, the full block–tridiagonal matrix for the 3D case is very similar to that for 2D, as shown in Figure 11. The off–diagonal matrix blocks are banded as in the 2D case, only the bands are not as structured.

It must be kept in mind, that this is merely one possible arrangement of the data points lying on each diagonal plane. Indications are that virtually all consistent sampling methods result in systems which look a great deal like this. This particular technique of using diagonal plane ordering seems to reduce the size of the off–diagonal matrix–blocks to a minimum. This is an important consideration when dealing with any form of solver in which matrix fill–in will occur.

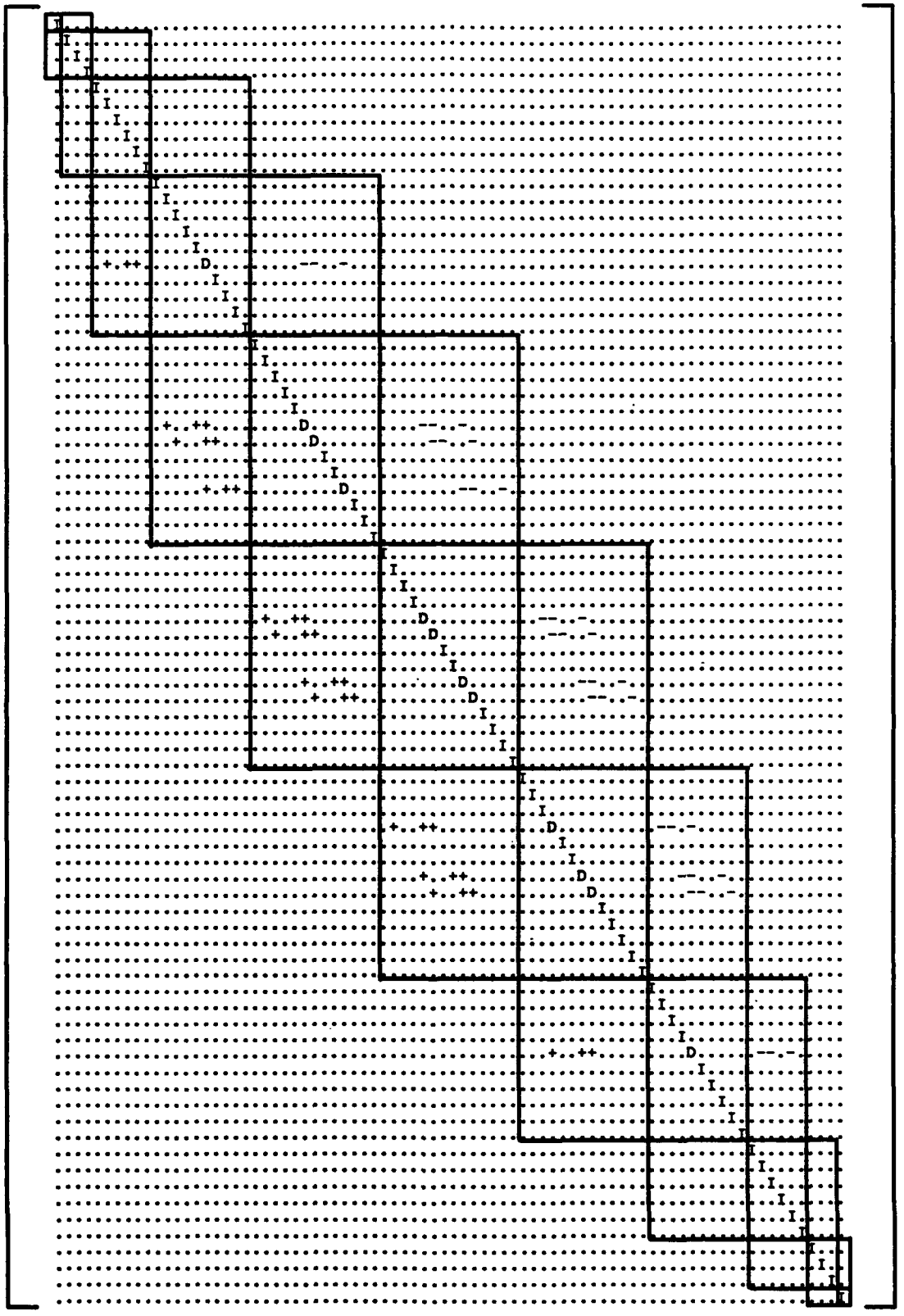


Figure 11. Three–Dimensional System Matrix Structure Using Diagonal Plane Ordering

## 2. OPTIMIZING APPROXIMATE FACTORIZATION

In the last subsection, it was shown that the system matrix resulting from the three–dimensional fluid dynamic equations can be ordered such that it appears as a quasi–block–tridiagonal system. At this point, a direct solution technique which exploits this block matrix structure could be implemented for the inversion of the system matrix. Alternately, approximate factorization (AF) is a solution method which has been advocated by the authors for sometime now. Essentially AF breaks the M system matrix into upper and lower triangular matrices much the same as an LU decomposition (of a direct solver) would. The two–factor scheme (two–pass scheme of Section IV) put forth sometime ago yields an upper triangular matrix which includes all of the backward running information, and a lower triangular matrix which includes all forward running information. Since the original equation looks like $M\Delta Q = R$, then the resulting form will be $M^{+}M^{-}\Delta Q = R$. This is the same sort of equation that one gets from an LU decomposition of a direct (block–tridiagonal) solver, and can be solved using 5x5 block matrix inversion coupled with backward (forward) substitution. What is sought here is to understand the "communication" penalty associated with approximating the LU decomposition one would get from a direct solver. Initially, consider the block–tridiagonal matrix structure resulting from the one–dimensional equations as discussed previously. Expanding the one–dimensional equation for an arbitrary cell i yields,

$$( I + A_i^{+} - A_{i-1}^{+} + A_{i+1}^{-} - A_i^{-} ) \Delta Q = R \tag{77}$$

where I is a 3x3 identity matrix, the A's are 3x3 flux Jacobians and $\Delta Q$ and R are the solution vector and residual, respectively. This equation is used for all cells in the domain, resulting in a system of equations for simultaneous solution. In matrix form the equation obtained is

$$M\ \Delta Q = R \tag{78}$$

where

$$M = \begin{bmatrix} M_{11} & M_{12} & 0 & 0 & \cdots \\ M_{21} & M_{22} & M_{23} & 0 & \cdots \\ 0 & M_{32} & M_{33} & M_{34} & \cdots \\ 0 & 0 & M_{43} & M_{44} & \cdots \\ 0 & 0 & 0 & M_{54} & \cdots \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

and $\Delta Q$ and R are column vectors.

A standard block–tridiagonal LU decomposition results in the following matrix structure

$$M = L\,U = \begin{bmatrix} L_{11} & 0 & 0 & 0 & \cdots \\ L_{21} & L_{22} & 0 & 0 & \cdots \\ 0 & L_{32} & L_{33} & 0 & \cdots \\ 0 & 0 & L_{43} & L_{44} & \cdots \\ 0 & 0 & 0 & L_{54} & \cdots \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} & 0 & 0 & \cdots \\ 0 & U_{22} & U_{23} & 0 & \cdots \\ 0 & 0 & U_{33} & U_{34} & \cdots \\ 0 & 0 & 0 & U_{44} & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{79}$$

The first few entries in the M matrix are given by,

$$\begin{aligned} M_{11} &= L_{11}U_{11} = I + A_1^+ - A_1^- \\ M_{12} &= L_{11}U_{12} = A_2^- \\ M_{21} &= L_{21}U_{11} = -A_1^+ \\ M_{22} &= L_{22}U_{22} + L_{21}U_{12} = I + A_2^+ - A_2^- \\ M_{23} &= L_{22}U_{23} = A_3^- \\ M_{32} &= L_{32}U_{22} = -A_2^+ \\ M_{33} &= L_{33}U_{33} + L_{32}U_{23} = I + A_3^+ - A_3^- \\ M_{34} &= L_{33}U_{34} = A_4^- \\ &\vdots \end{aligned} \tag{80}$$

For the standard two–factor scheme, the operator of Equation (77) is approximately factored as follows,

$$\begin{aligned} (I + A_i^+ - A_{i-1}^+)\,(I + A_{i+1}^- - A_i^-)\,\Delta Q &= R \\ L\,U\,\Delta Q &= R \end{aligned} \tag{81}$$

The solution procedure for this two–factor scheme is

$$\begin{aligned} L\,X^1 &= R \\ U\,X^2 &= X^1 \\ \Delta Q = X^2 \quad &\Rightarrow \quad Q^{n+1} = Q^n + \Delta Q \end{aligned} \tag{82}$$

The structure for the approximate lower block triangular and upper block triangular matrices is

$$M \approx LU = \begin{bmatrix} I + A^+ & 0 & 0 & 0 & \cdots \\ -A^+ & I + A^+ & 0 & 0 & \cdots \\ 0 & -A^+ & I + A^+ & 0 & \cdots \\ 0 & 0 & -A^+ & I + A^+ & \cdots \\ 0 & 0 & 0 & -A^+ & \cdots \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} I - A^- & A^- & 0 & 0 & \cdots \\ 0 & I - A^- & A^- & 0 & \cdots \\ 0 & 0 & I - A^- & A^- & \cdots \\ 0 & 0 & 0 & I - A^- & \cdots \\ 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{83}$$

which leads to the following entries in the resulting approximate M matrix (operator product matrix),

$$
\begin{aligned}
M_{11} &= L_{11}U_{11} = I + A_1^+ - A_1^- - \underline{A_1^+}\,\underline{A_1^-} \\
M_{12} &= L_{11}U_{12} = A_2^- + \underline{A_1^+}\,\underline{A_2^-} \\
M_{21} &= L_{21}U_{11} = -A_1^+ + \underline{A_1^+}\,\underline{A_1^-} \\
M_{22} &= L_{22}U_{22} + L_{21}U_{12} = I + A_2^+ - A_2^- - \underline{A_2^+}\,\underline{A_2^-} - \underline{A_1^+}\,\underline{A_2^-} \\
M_{23} &= L_{22}U_{23} = A_3^- + \underline{A_2^+}\,\underline{A_3^-} \\
M_{32} &= L_{32}U_{22} = -A_1^+ + \underline{A_2^+}\,\underline{A_2^-} \\
M_{33} &= L_{33}U_{33} + L_{32}U_{23} = I + A_3^+ - A_3^- - \underline{A_3^+}\,\underline{A_3^-} - \underline{A_2^+}\,\underline{A_3^-} \\
M_{34} &= L_{33}U_{34} = A_4^- + \underline{A_3^+}\,\underline{A_4^-} \\
&\;\;\vdots
\end{aligned}
\tag{84}
$$

The terms which are underlined do not appear in the solution matrix of the direct solver and hence are the AF errors. Note how they are dispersed throughout the matrix.

Next, take a look at the modified two–factor scheme (modified two–pass scheme of Section IV). Rewriting Equation (77) as

$$( D_i - A_{i-1}^+ + A_{i+1}^- ) \Delta Q = R$$

where

$$D_i = I + A_i^+ - A_i^-$$

The modified two–factor scheme is given by the following approximate factoring

$$( D_i - A_{i-1}^+ ) D_i^{-1} ( D_i + A_{i+1}^- ) \Delta Q = R \tag{85}$$

thus

$$L\,D\,U\,\Delta Q = R$$

The solution procedure for this modified two–factor scheme can be viewed as follows

$$
\begin{aligned}
L\,X^1 &= R \\
D\,X^2 &= X^1 \\
U\,X^3 &= X^2 \\
\Delta Q = X^3 \quad &\Rightarrow \quad Q^{n+1} = Q^n + \Delta Q
\end{aligned}
\tag{86}
$$

The structure for the approximate lower block triangular and upper block triangular matrices is

$$M \approx LDU = \begin{bmatrix} L_{11} & 0 & 0 & 0 & \\ L_{21} & L_{22} & 0 & 0 & \\ 0 & L_{32} & L_{33} & 0 & \\ 0 & 0 & L_{43} & L_{44} & \\ 0 & 0 & 0 & L_{54} & \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} D \begin{bmatrix} U_{11} & U_{12} & 0 & 0 & \\ 0 & U_{22} & U_{23} & 0 & \\ 0 & 0 & U_{33} & U_{34} & \\ 0 & 0 & 0 & U_{44} & \\ 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{87}$$

To lessen the confusion between the system matrix D and the point–block matrices denoted $D_i$ for each cell, define

$$L' = L\,D = \begin{bmatrix} I & 0 & 0 & 0 & \\ -A_1^+ D_1^{-1} & I & 0 & 0 & \\ 0 & -A_2^+ D_2^{-1} & I & 0 & \\ 0 & 0 & -A_3^+ D_3^{-1} & I & \\ 0 & 0 & 0 & -A_4^+ D_4^{-1} & \\ 0 & 0 & 0 & 0 & \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{88}$$

thus,

$$M \approx L'U \tag{89}$$

which leads to the following entries in the resulting approximate M matrix (operator product matrix),

$$\begin{aligned}
M_{11} &= L'_{11}U_{11} = D_1 = I + A_1^+ - A_1^- \\
M_{12} &= L'_{11}U_{12} = A_2^- \\
M_{21} &= L'_{21}U_{11} = -A_1^+ D_1^{-1} D_1 = -A_1^+ \\
M_{22} &= L'_{22}U_{22} + L_{21}U_{12} = I + A_2^+ - A_2^- - \underline{A_1^+ D_1^{-1} A_2^-} \\
M_{23} &= L'_{22}U_{23} = A_3^- \\
M_{32} &= L'_{32}U_{22} = -A_1^+ \\
M_{33} &= L'_{33}U_{33} + L_{32}U_{23} = I + A_3^+ - A_3^- - \underline{A_2^+ D_2^{-1} A_3^-} \\
M_{34} &= L'_{33}U_{34} = A_4^- \\
&\vdots
\end{aligned} \tag{90}$$

The terms appearing which are underlined are again the AF errors. As can be seen, for the modified two–factor scheme the error terms only appear in the diagonal elements. Although presently just a speculative observation, it would appear that the error associated with the modified two–factor scheme is far less than that of the standard two–factor scheme. Defining the block matrices $D_i$ as follows

$$D_i = I + A_i^+ - A_i^- + A_{i-1}^+ D_{i-1}^{-1} A_i^- \qquad (91)$$

all the AF error is removed yielding a block-tridiagonal matrix direct solution procedure (Thomas' algorithm). This works well in one-dimension when the blocks are full, there's no memory penalty. With a single dimension, the system matrix is a matrix with block elements each of which is composed of 3x3 scalar entries (point-blocks). In this simple case of the one-dimensional system all of these elements (point-blocks) are full, i.e. the diagonal and off-diagonal elements (point-blocks) are full 3x3 sub-matrices. In multiple dimensions there's one caveat. For multiple dimensions, the system matrix is a matrix with block elements (matrix-blocks) which are in turn composed of block matrices (point-blocks) which have NxN scalar entries, where N is the number of dependent variables at each point. The diagonal and off-diagonal matrix-blocks are sparse block-banded sub-matrices. It is this sparsity which is desired for an efficient solution procedure. As a matter of fact the diagonal structure of the diagonal block (when using a matrix ordering based on diagonal plane processing) is what provides the vector processing ability for the AF methods. The lack of off-diagonal point-blocks in the diagonal matrix-block represents the local uncoupled nature of points lying on diagonal planes, setting the stage for simultaneous (vector) processing. The point (2,2), for example, is influenced by the point (3,1) indirectly. With a direct solver, during the solution procedure, matrix fill-in creates an influence coefficient point-block matrix (communication path) where previously there was none (null point-block). These paths are indirect as is indicated by the wide open arrows in Figure 12, and are very numerous since all points will influence all others to some extent.

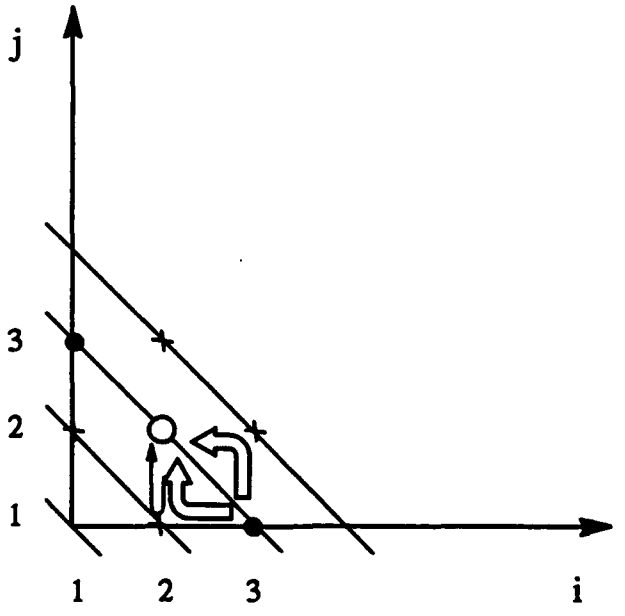



Figure 12. Indirect Communication Paths

In the figure colinear (diagonal) points are shown with a dark dot, the (haloed) point of interest is in direct communication with the points to the immediate right and left, and top and bottom. Colinear neighboring points are only in indirect communication. In addition, the impact of the solution of (2,2) on, for example, (2,1) will again impact (2,2) the path of which is indicated with the solid arrow.

Examining the two–dimensional case a little further will reveal more details of this dilemma. The modified two–factor scheme written for two dimensions is

$$( D_{i,j} - A^{+}_{i-1,j} - B^{+}_{i,j-1} ) D^{-1}_{i,j} ( D_{i,j} + A^{-}_{i+1,j} + B^{-}_{i,j+1} ) \Delta Q = R \qquad (92)$$

where

$$D_{i,j} = I + A^{+}_{i,j} - A^{-}_{i,j} + B^{+}_{i,j} - B^{-}_{i,j} \qquad (93)$$

Consider the system matrix resulting from a diagonal line ordering.

$$M = \begin{array}{c} \begin{array}{cccccccccc} 1,1 & 1,2 & 2,1 & 1,3 & 2,2 & 3,1 & 1,4 & 2,3 & 3,2 & 4,1 \end{array} \\ \left[\begin{array}{cccccccccc} D & B^{-} & A^{-} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -B^{+} & D & 0 & B^{-} & A^{-} & 0 & 0 & 0 & 0 & 0 \\ -A^{+} & 0 & D & 0 & B^{-} & A^{-} & 0 & 0 & 0 & 0 \\ 0 & -B^{+} & 0 & D & 0 & 0 & B^{-} & A^{-} & 0 & 0 \\ 0 & -A^{+} & -B^{+} & 0 & D & 0 & 0 & B^{-} & A^{-} & 0 \\ 0 & 0 & -A^{+} & 0 & 0 & D & 0 & 0 & B^{-} & A^{-} \\ 0 & 0 & 0 & -B^{+} & 0 & 0 & D & 0 & 0 & 0 \\ 0 & 0 & 0 & -A^{+} & -B^{+} & 0 & 0 & D & 0 & 0 \\ 0 & 0 & 0 & 0 & -A^{+} & -B^{+} & 0 & 0 & D & 0 \\ 0 & 0 & 0 & 0 & 0 & -A^{+} & 0 & 0 & 0 & D \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right] \end{array} \begin{array}{c} 1,1 \\ 1,2 \\ 2,1 \\ 1,3 \\ 2,2 \\ 3,1 \\ 1,4 \\ 2,3 \\ 3,2 \\ 4,1 \\ \vdots \end{array} \qquad (94)$$

The diagonal matrix–blocks are diagonal themselves, hence the local uncoupled nature of the points lying on diagonal lines is readily seen. The question then arises, "How do they get coupled?", because it is known that for an implicit scheme, such as that used here, all points influence all other points. It is this coupling which differentiates the direct solver from the approximation techniques. For a direct solver during the LU decomposition process the one of the diagonal matrix–blocks belonging to either the lower or upper matrix will no longer retain this diagonal structure, it fills in. This fill–in accomplishes the coupling between points lying on a diagonal line (or plane in 3D) but ruins any vectorization along diagonal lines (or planes in 3D). If the retension of no fill–in and the current approach for vector processing is desired, then some data paths (solution coupling) must be cut. The modified two–factor seems to have done an excellent job of reconnecting data paths cut by the standard two–factor and/or disconnecting faulty data paths created by the standard two–factor scheme in the off–diagonal matrix–blocks and some in diagonal matrix–blocks. There exists some AF error (data path error) which re-

mains in the diagonal matrix–blocks which can be removed while still retaining the no fill–in matrix structure enjoyed with the AF schemes. To see this, one can simply run through the matrix products which result from the AF process and compare to the matrix in Equation (94) above. Consider the following AF matrices

$$L' = \begin{array}{c} \begin{array}{cccccc} 1,1 & 1,2 & 2,1 & 1,3 & 2,2 & 3,1 \end{array} \\ \left[\begin{array}{|c|cc|ccc|c} \hline I & 0 & 0 & 0 & 0 & 0 & \\ \hline -B^{+}D^{-1} & I & 0 & 0 & 0 & 0 & \\ -A^{+}D^{-1} & 0 & I & 0 & 0 & 0 & \\ \hline 0 & -B^{+}D^{-1} & 0 & I & 0 & 0 & \\ 0 & -A^{+}D^{-1} & -B^{+}D^{-} & 0 & I & 0 & \\ 0 & 0 & -A^{+}D^{-} & 0 & 0 & I & \\ \hline \end{array}\right] \end{array} \begin{array}{c} 1,1 \\ 1,2 \\ 2,1 \\ 1,3 \\ 2,2 \\ 3,1 \end{array} \tag{95}$$

and

$$U = \begin{array}{c} \begin{array}{cccccc} 1,1 & 1,2 & 2,1 & 1,3 & 2,2 & 3,1 \end{array} \\ \left[\begin{array}{|ccc|ccc|c} \hline D & B^{-} & A^{-} & 0 & 0 & 0 & \\ 0 & D & 0 & B^{-} & A^{-} & 0 & \\ 0 & 0 & D & 0 & B^{-} & A^{-} & \\ \hline 0 & 0 & 0 & D & 0 & 0 & \\ 0 & 0 & 0 & 0 & D & 0 & \\ 0 & 0 & 0 & 0 & 0 & D & \\ \hline \end{array}\right] \end{array} \begin{array}{c} 1,1 \\ 1,2 \\ 2,1 \\ 1,3 \\ 2,2 \\ 3,1 \end{array} \tag{96}$$

Using row–column subscripts (with no commas) to denote matrix–blocks, the first few entries in the resulting system matrix are

$$\begin{aligned} M_{11} &= L'_{11}\, U_{11} = I\, D_{1,1} = \left[\, D_{1,1} \,\right] \\ M_{12} &= L'_{11}\, U_{12} = I \left[\, B^{-}_{1,2}\; A^{-}_{2,1} \,\right] = \left[\, B^{-}_{1,2}\; A^{-}_{2,1} \,\right] \\ M_{21} &= L'_{21}\, U_{11} = \begin{bmatrix} (-B^{+}_{1,1}\, D^{-1}_{1,1}) \\ (-A^{+}_{1,1}\, D^{-1}_{1,1}) \end{bmatrix} D_{1,1} = \begin{bmatrix} -B^{+}_{1,1} \\ -A^{+}_{1,1} \end{bmatrix} \\ M_{22} &= L'_{22}\, U_{22} + L'_{21}\, U_{12} = \begin{bmatrix} D_{1,2} & 0 \\ 0 & D_{2,1} \end{bmatrix} - \begin{bmatrix} B^{+}_{1,1}\, D^{-1}_{1,1}\, B^{-}_{1,2} & B^{+}_{1,1}\, D^{-1}_{1,1}\, A^{-}_{2,1} \\ A^{+}_{1,1}\, D^{-1}_{1,1}\, B^{-}_{1,2} & A^{+}_{1,1}\, D^{-1}_{1,1}\, A^{-}_{2,1} \end{bmatrix} \\ &\vdots \end{aligned} \tag{97}$$

Examining the last entry shown here, it can be seen that the AF error appears in a matrix–block in the form of point–blocks. The off–diagonal entries in this error matrix–block can not be removed if no fill–in is desired, but the errors appearing on the diagonal can be removed by

adding those terms to the point–block $D_{i,j}$. As long as the diagonal matrix–blocks are maintained as diagonal block matrices themselves, there is no fill–in. Thus the point–block $D_{i,j}$ is defined

$$D_{i,j} = I + A^{+}_{i,j} - A^{-}_{i,j} + B^{+}_{i,j} - B^{-}_{i,j} + A^{+}_{i-1,j}D^{-1}_{i-1,j}A^{-}_{i,j} + B^{+}_{i,j-1}D^{-1}_{i,j-1}B^{-}_{i,j} \quad (98)$$

or

$$D_{i,j} = I + A^{+}_{i,j} + B^{+}_{i,j} + ( A^{+}_{i-1,j}D^{-1}_{i-1,j} - I ) A^{-}_{i,j} + ( B^{+}_{i,j-1}D^{-1}_{i,j-1} - I ) B^{-}_{i,j} \quad (99)$$

for optimal solution coupling with no fill–in. The extension of this to three dimensions is straight forward, yielding

$$\begin{aligned} D_{i,j,k} = I &+ A^{+}_{i,j,k} + B^{+}_{i,j,k} + C^{+}_{i,j,k} + ( A^{+}_{i-1,j,k}D^{-1}_{i-1,j,k} - I ) A^{-}_{i,j,k} \\ &+ ( B^{+}_{i,j-1,k}D^{-1}_{i,j-1,k} - I ) B^{-}_{i,j,k} + ( C^{+}_{i,j,k-1}D^{-1}_{i,j,k-1} - I ) C^{-}_{i,j,k} \end{aligned} \quad (100)$$

To test the new factoring scheme, a simple two–dimensional geometry was selected. The geometry is that of a NACA 64A012 airfoil at two degrees angle–of–attack in a Mach 0.87 flow. An H–grid using two blocks (each 71 x 31 ), one upper and one lower was used to discretize the domain. Both the upper and lower surfaces of the airfoil are modeled with thirty cells.

The results of the tests are given in Figures 13 and 14 . These figures show a comparison of the convergence histories for the standard two–factor (STDAF), the modified two–factor (MAF), and the optimized two–factor (OAF). It can be seen from the limited tests that were run (using local time–stepping) that the sensitivity to larger CFL numbers exhibited by the STDAF scheme is not apparent with either MAF or OAF. Although this is the case, for this test configuration, both MAF and OAF seem to exhibit a less rapid convergence rate than the STDAF for a given CFL number. It is important to note that, although not shown, both MAF and OAF maintained stable convergence for CFL's in the hundreds while the STDAF went divergent early–on for a CFL of one hundred. Little difference in convergence rate is seen between the MAF and the OAF for this particular configuration. So little difference in fact that the extra computations necessary in the OAF method are not warranted. This may not be the case for some other geometries though. More convergence analyses need to be performed on various geometries and flow conditions.

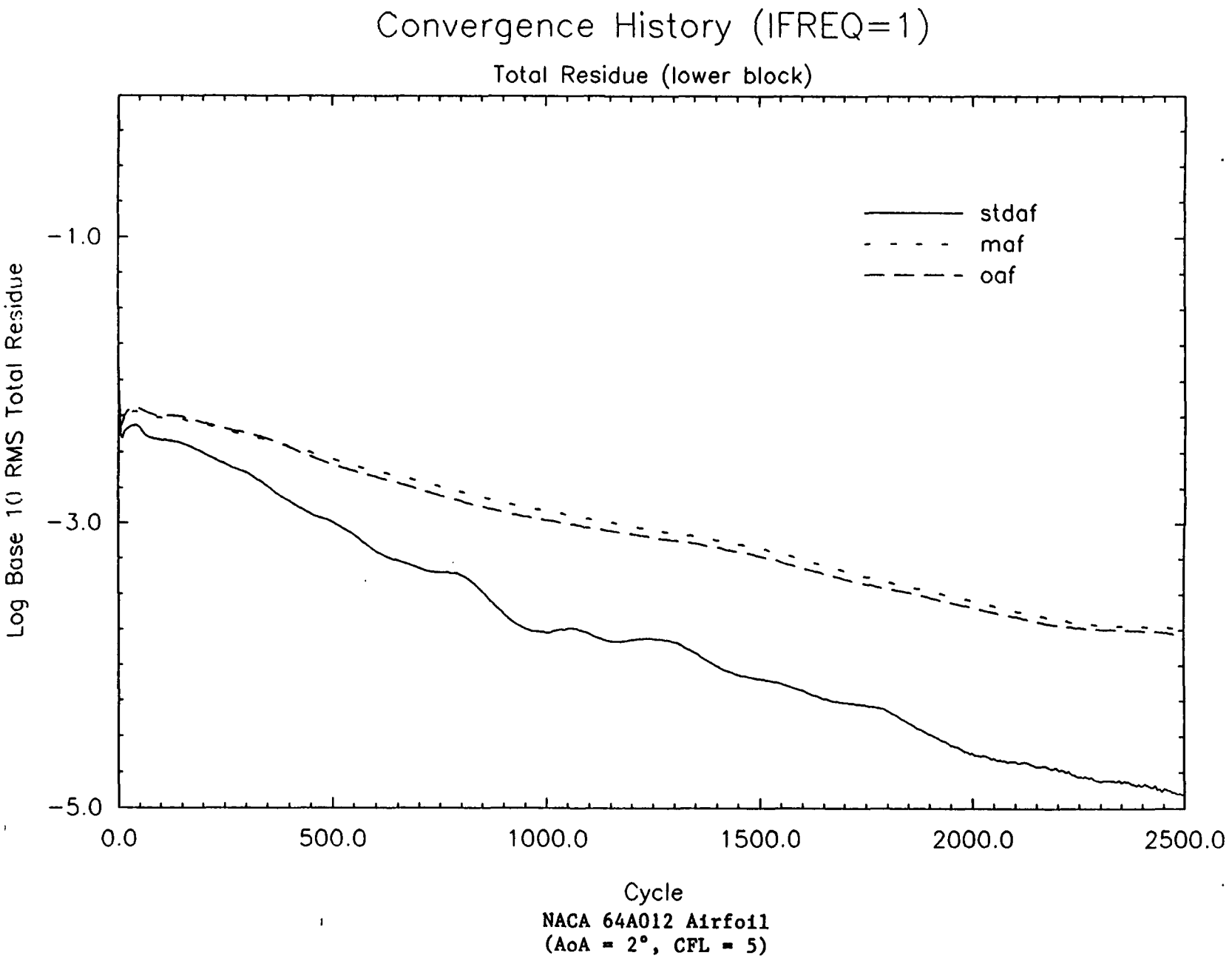



**Figure 13a. Approximate Factorization Operator Convergence History Comparison (CFL = 5, Lower Block)**

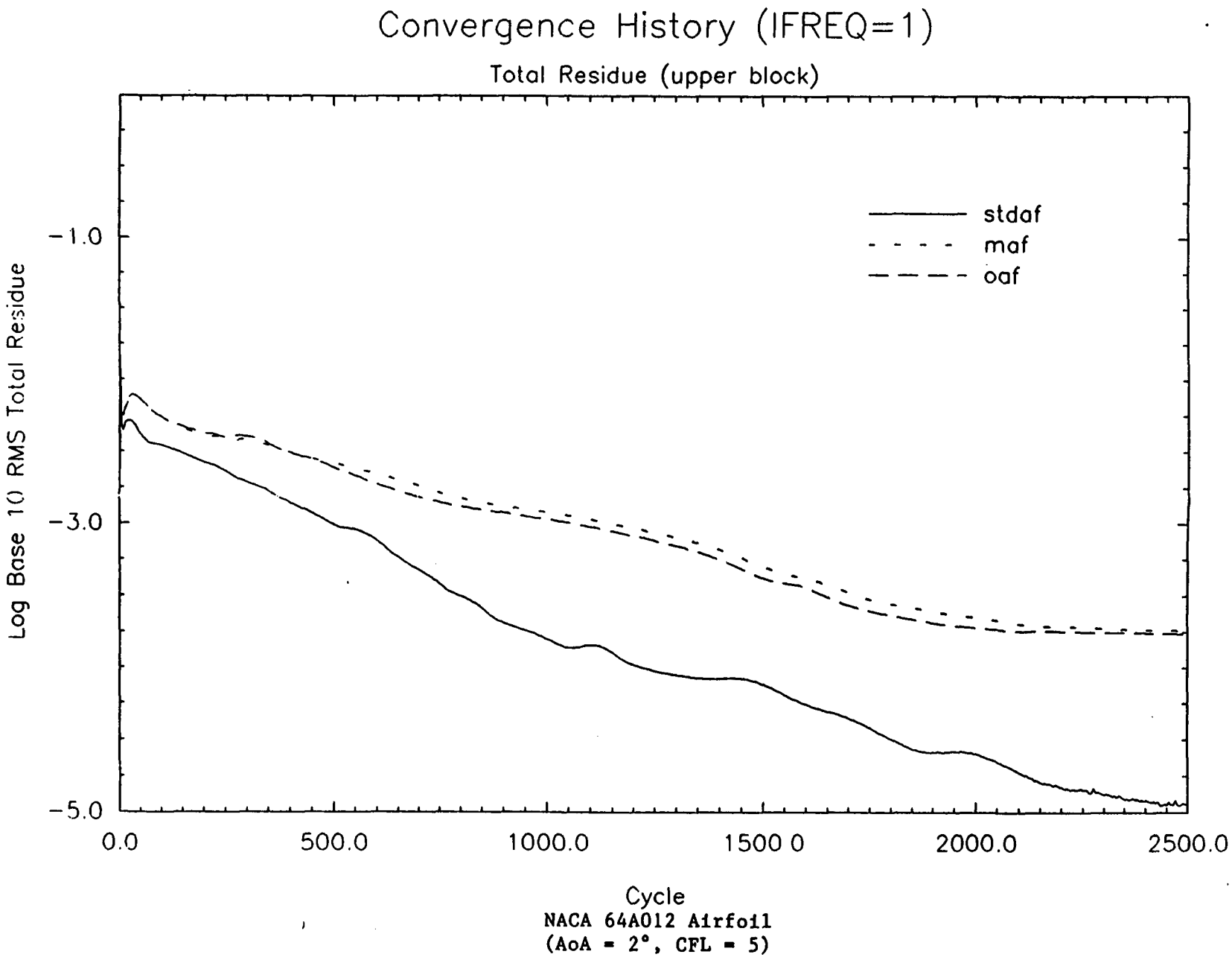



**Figure 13b. Approximate Factorization Operator Convergence History Comparison (CFL = 5, Upper Block)**

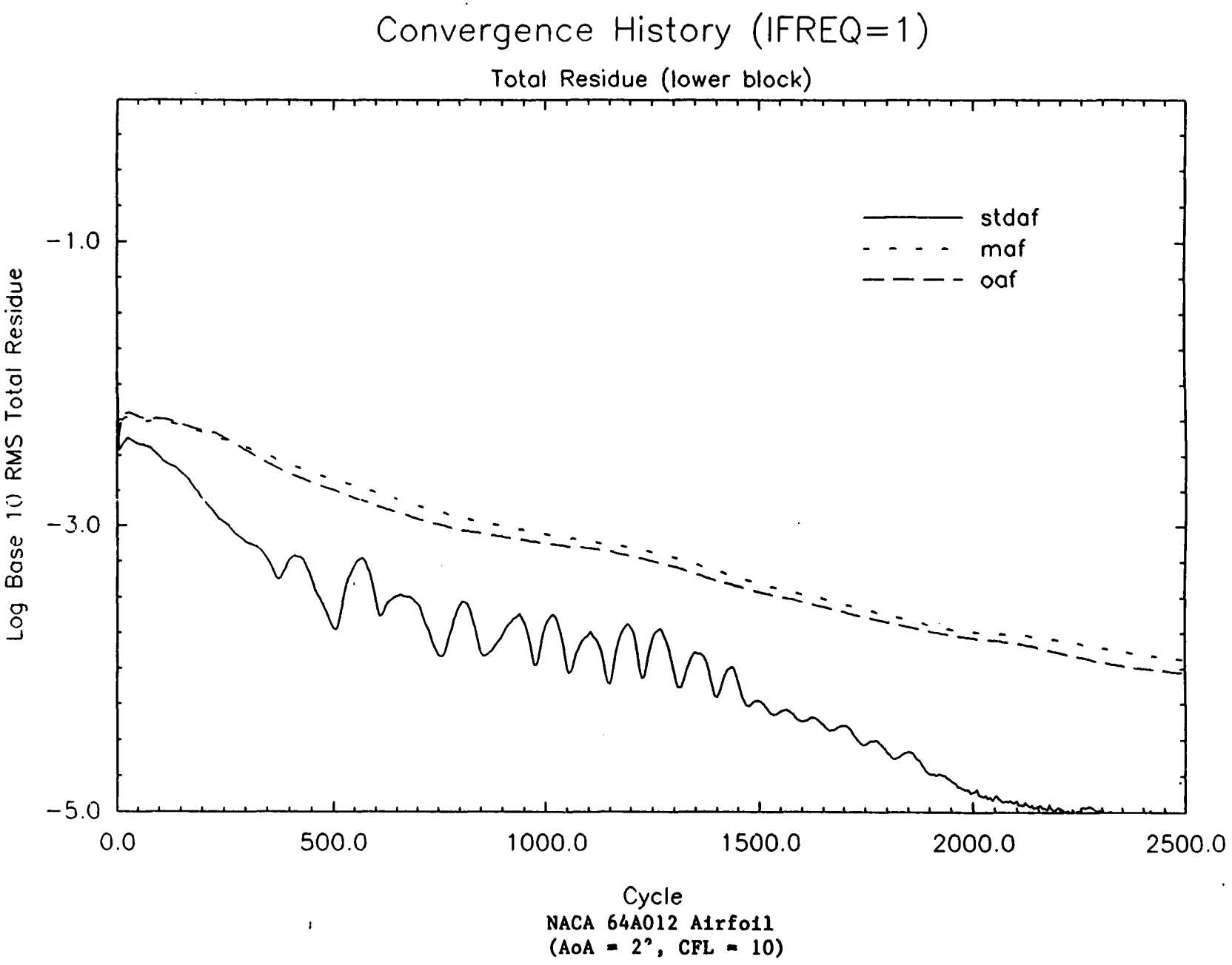



**Figure 14a. Approximate Factorization Operator Convergence History Comparison (CFL = 10, Lower Block)**

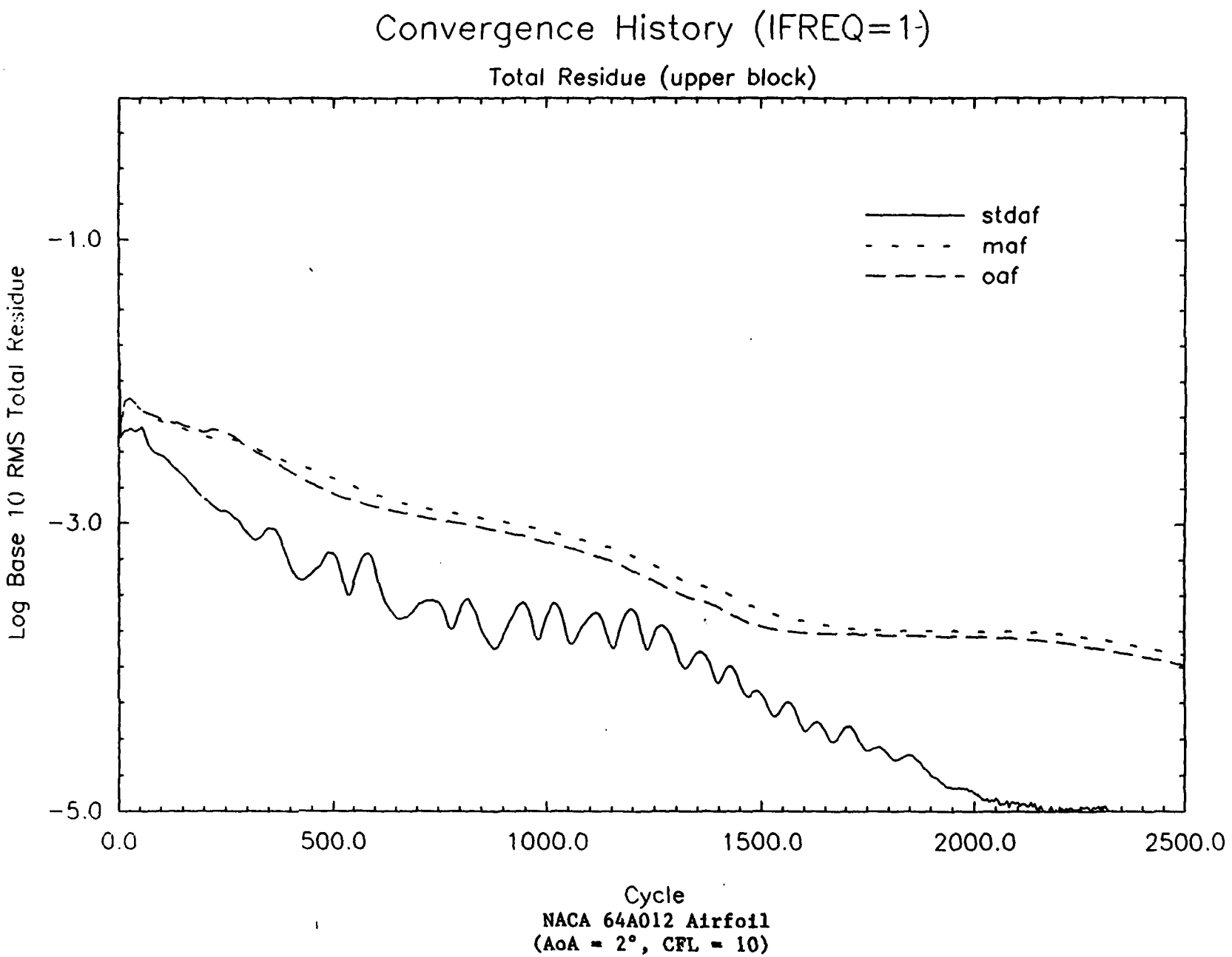



**Figure 14b. Approximate Factorization Operator Convergence History Comparison (CFL = 10, Upper Block)**

## 3. DIAGONAL PLANE PROCESSING SOFTWARE ANALYSIS

During the course of the software (and algorithm) development for the flow model presented here significant emphasis has been placed on the impact of solution vectorization. So much in fact that special matrix orderings were selected to "create" computational vectors for simultaneous solution. This has proven to be a viable technique although there is one known shortcoming, data latency. Cray vector processors store data in a sequence of memory banks. Each machine is different in the number of memory banks it contains, usually some multiple of eight, i.e. 8, 16, 32, etc. The memory also has a specific reserved access time, referred to as the bankbusy cycle time which is measured in clock cycles. If a memory bank is accessed twice within the bankbusy cycle time the cpu must wait for the data requested. This situation is referred to as a memory bank conflict and can significantly disturb the processing rate of an otherwise standard vector loop, see Reference 30. Conditions that create memory bank conflicts which should be avoided for standard (regular stride) loops are well documented in the Cray manuals. The difficulty here is the loops associated with diagonal plane processing use indirect addressing which results in an uneven stride and hence the potential for unpredictable memory bank conflicts.

To examine this, a test case was devised which could have the computational domain resized automatically (max i, j, k indices varied) such that all typical grid block dimensions could be tested for memory bank conflicts on a particular machine. A grids block dimensions determine the nature of the array storage in the memory banks and hence by parametrically timing the processing rate (of the matrix inversion using diagonal plane processing) of different dimension blocks, certain trends can be observed to steer a user away from a particular grid size. To date this has been quite a successful undertaking. A parametric analysis of grid sizes ranging from $10 < i_{max} < 66$, $4 < j_{max} < 32$, and $4 < k_{max} < 24$ has been accomplished on a Cray X–MP with 32 memory banks and a bankbusy value of 8 clock cycles. What was observed from this study was that grid dimensions involving an $i_{max}$ of (multiples of 8) + 1 created significantly more memory bank conflicts than the other grids regardless of the $j_{max}$ and $k_{max}$ values, see Figures 15. Grids with $i_{max}$ of 17, and 49 showing moderate increases in processing time, and 33, and 65 showing approximately a three– to four–fold increase in processing time. The complexity of the operations involved in the loops and the manner in which the Cray manipulates data prohibits (at the present time) a detailed explanation as to the source(s) of conflicts for these grids. An interesting potential corrective measure was devised though.

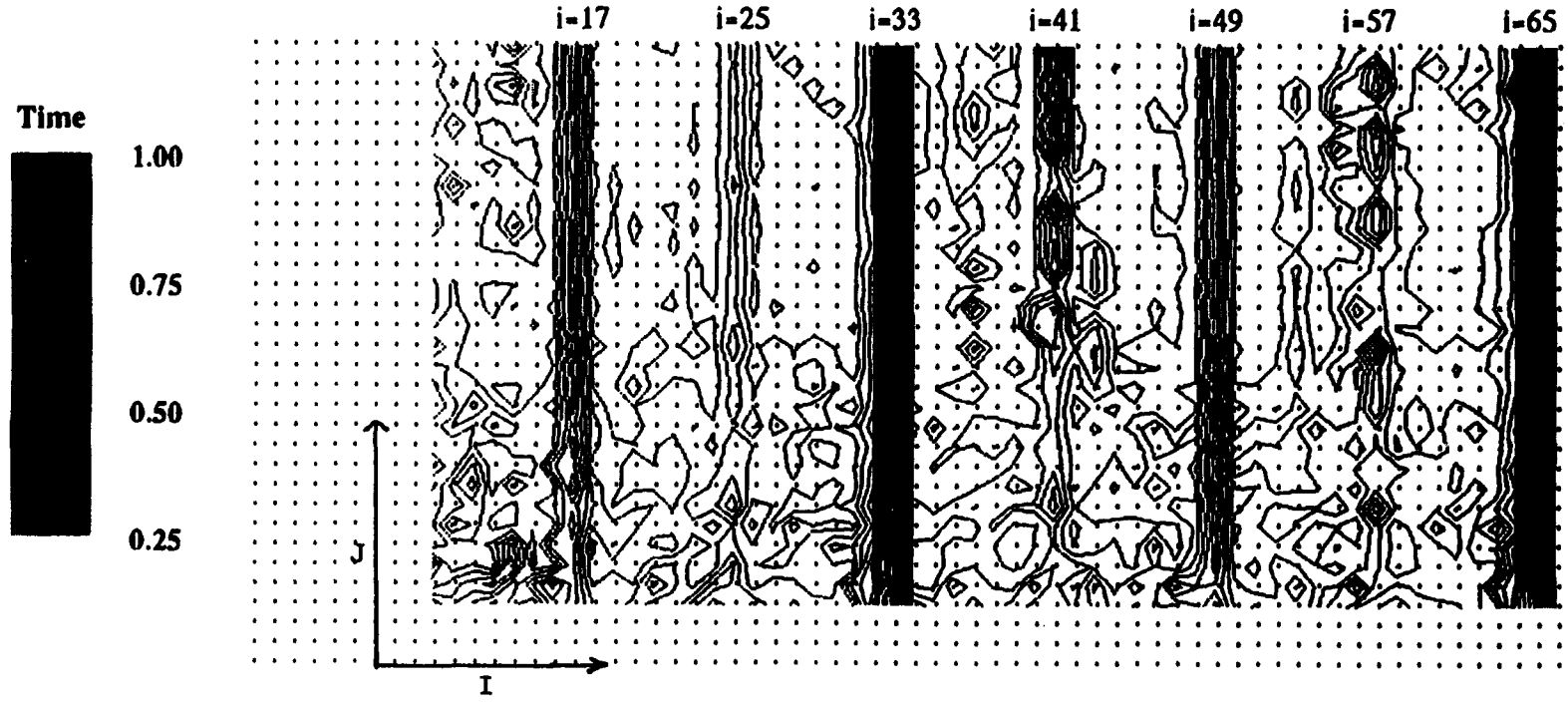



Figure 15a. Matrix Inversion CPU Time per Grid Point (K=10 Plane)

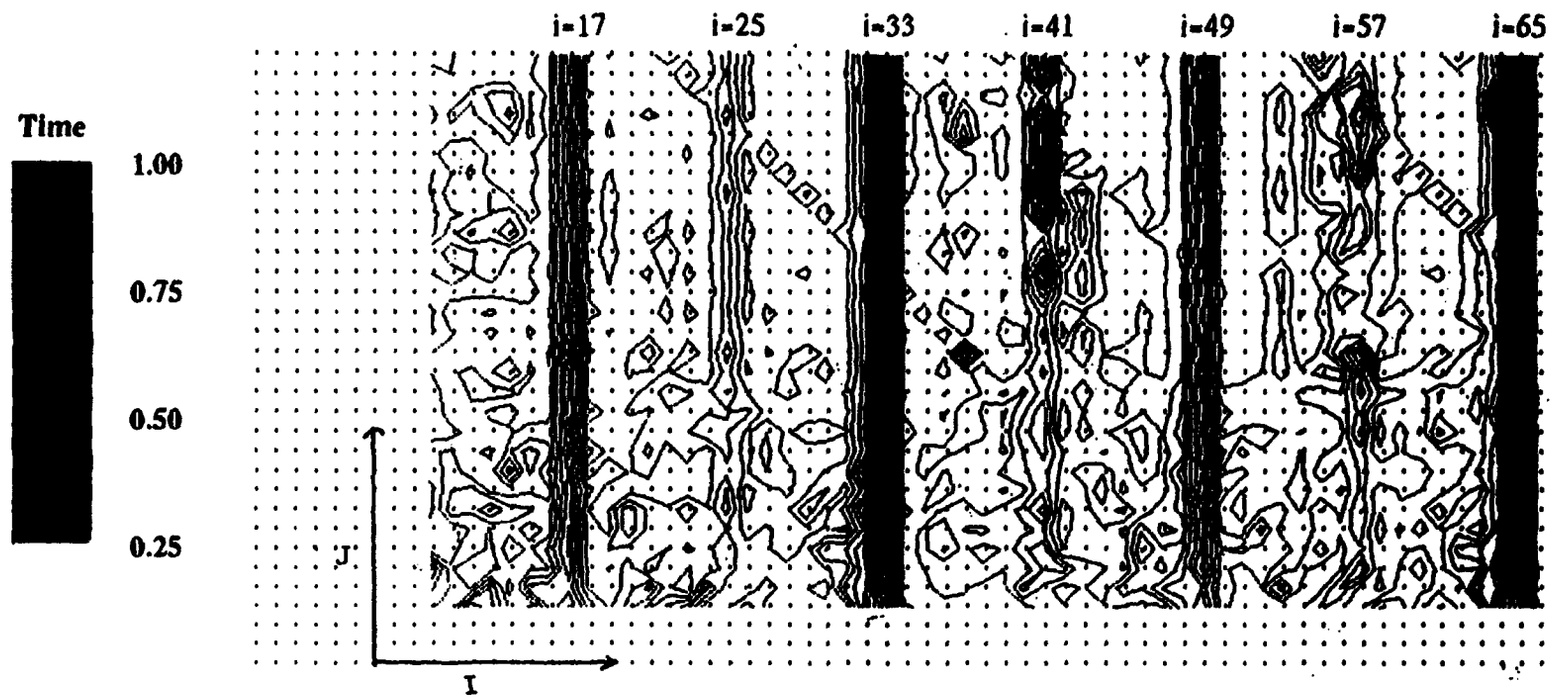



Figure 15b. Matrix Inversion CPU per Grid Point (K = 15 plane)

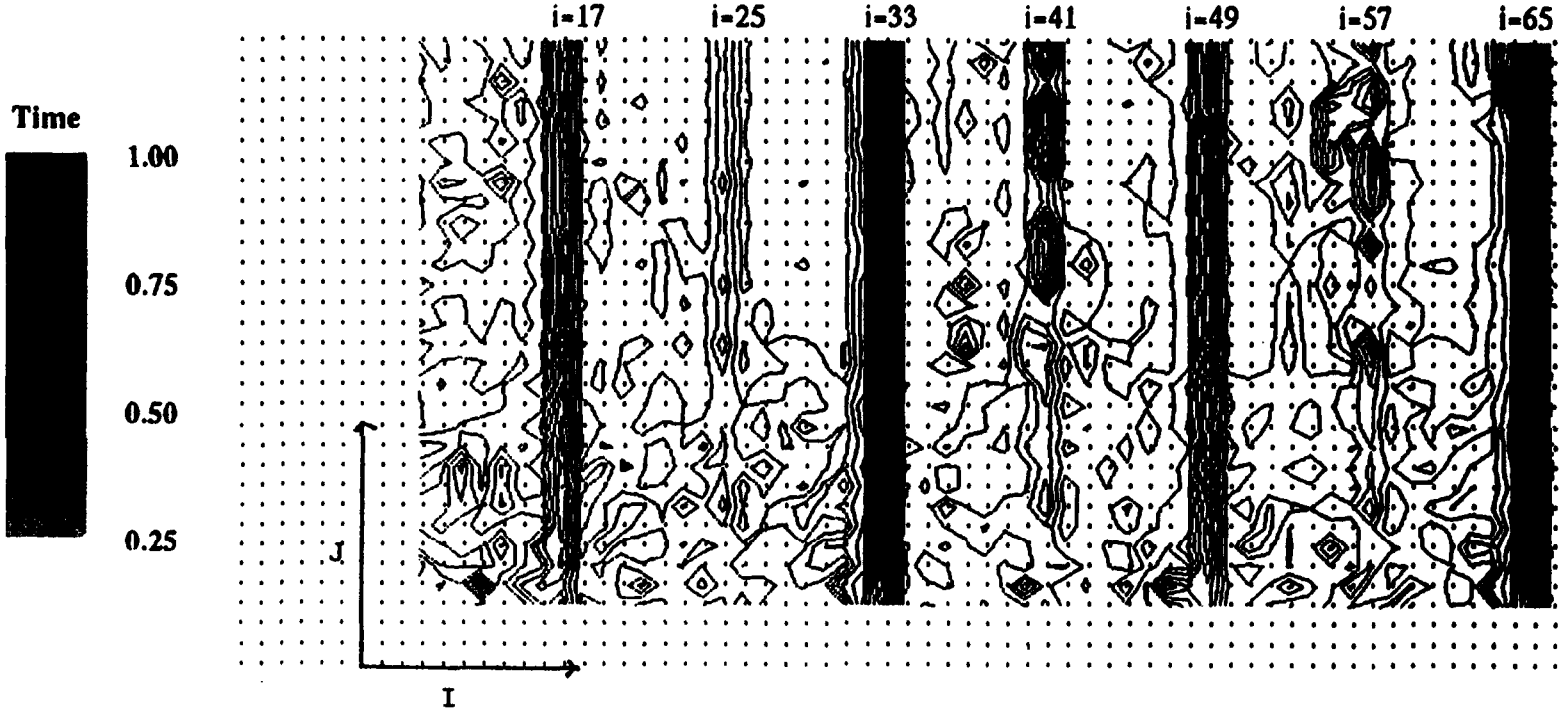
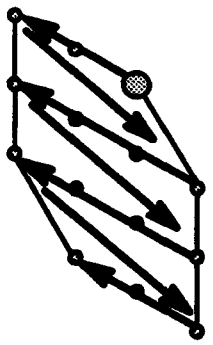



Figure 15c. Matrix Inversion CPU Time per Grid Point (K = 24 Plane)

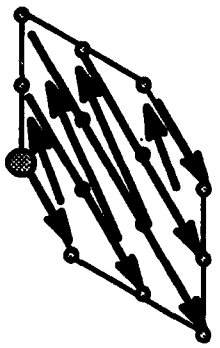
The manner in which data in the loops is accessed is determined largely by the mapping of the elements lying on the diagonal planes, i.e. the matrix-block structures. Coplanar solution nodes are computationally independent and hence the order in which they are grouped for processing is of no consequence to their simultaneous processing. The sequence of data acquisition on the other hand is greatly effected by this ordering. The simple mapping process described in Part 1 of this section was used in this study, i.e. KJI looping for a sampling routine. The interesting partial fix for grids with an $i_{max}$ dimension creating memory bank conflicts is to simply change the order in which the nodes are sampled to determine which plane they belong.

K J I looping



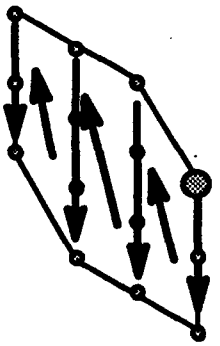
5,2,1
4,3,1
3,4,1
5,1,2
4,2,2
3,3,2
2,4,2
4,1,3
3,2,3
2,3,3
1.4.3
3,1,4
2,2,4
1,3,4

I K J looping



1,4,3
1,3,4
2,4,2
2,3,3
2,2,4
3,4,1
3,3,2
3,2,3
3,1,4
4,3,1
4,2,2
4,1,3
5,2,1
5,1,2

J K I looping



5,1,2
4,1,3
3,1,4
5,2,1
4,2,2
3,2,3
2,2,4
4,3,1
3,3,2
2,3,3
1,3,4
3,4,1
2,4,2
1,4,3

Figure 16. Alternate Diagonal Plane Mappings of Points Lying on Plane Six

Figure 16 shows three different mappings of the fourteen points which lie on plane number six of the sample domain of Part 1 of this section. The starting location for each different mapping is the large point. Each point mapping (order) is listed to the side of the diagram. To establish the point order simply follow the arrows. Present indications reveal that moving the i loop as the outermost loop of the sampling (mapping) process the memory bank conflict phenomena could

be altered. For example, the grid with $i_{max}$ of 33 which previously posed a significant memory bank problem could be fixed simply by shifting to IKJ looping in the mapping process. Figure 17 shows the processing time of a small window around the $i_{max}$ of 33 grid size. It is obvious that $i_{max}$ of 33 no longer experiences memory bank conflicts to the extent seen for KJI looping. Although now grids with $j_{max}$ indices of 17 are hindered with conflicts as well as $i_{max}$ of 32. Additional data must be collected to determine reliable corrective measures for averting memory bank conflicts.

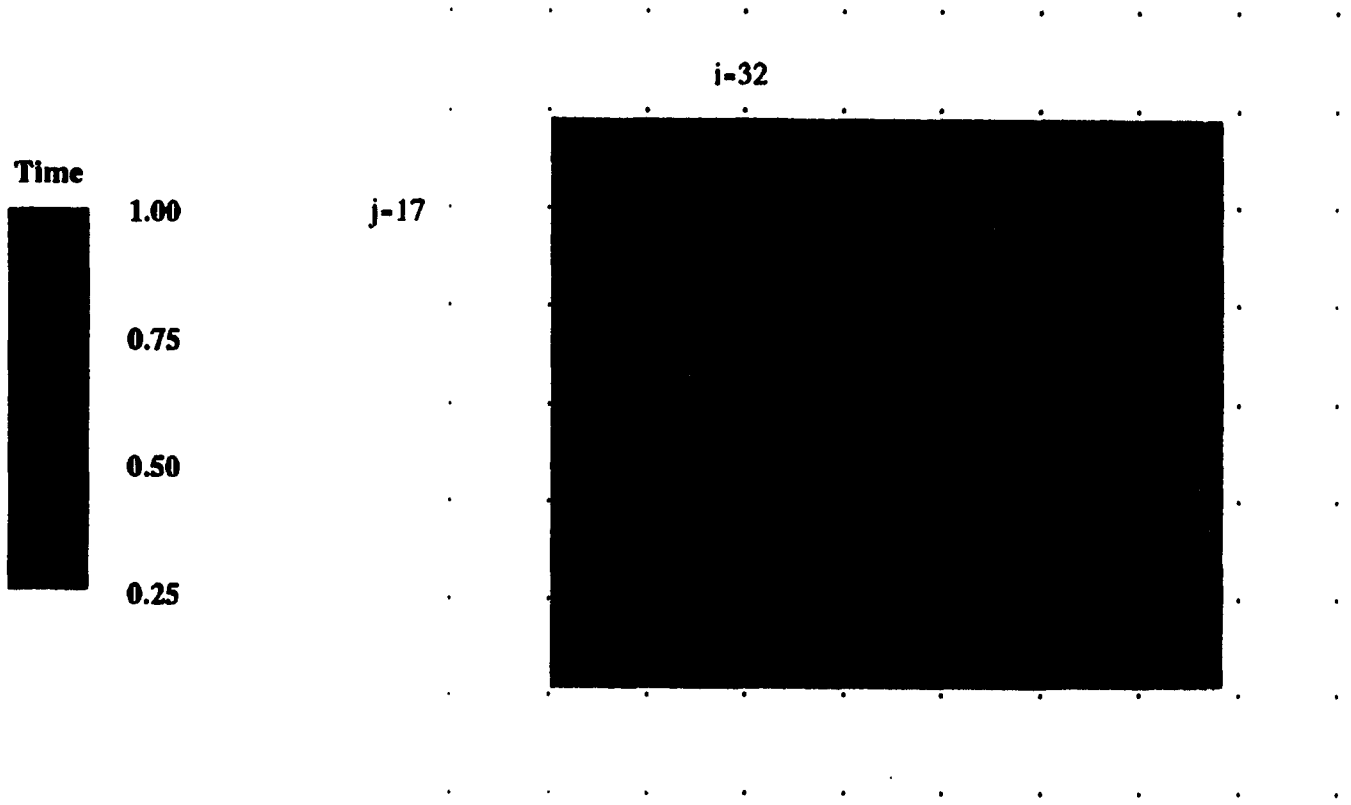



Figure 17. Matrix Inversion CPU Time per Grid Point (IKJ Looping, K-constant Plane)

## SECTION VI

## RESULTS AND DISCUSSION

### 1. HYPERSONIC BLUNT NOSE CYLINDER

Verification of the accuracy of any proposed numerical method of calculating the heat-transfer rate to the surface of a hypersonic re-entry vehicle invariably depends upon its subsequent agreement with experimental data. Therefore, in order to test and demonstrate the accuracy and efficiency of the present unsteady TLNS flow solver, the flow over two basic axisymmetric blunt-body configurations was investigated. The first configuration was a 15°-semiapex spherical, blunt cone with a bluntness ratio (ratio of nose radius to base radius) of 0.183. Model base diameter was one foot. The second configuration used in this investigation was a spherically capped, 9° half-angle cone. A C-type grid system is used to discretize the computation domain about these configurations. The second-order flux-difference split scheme with Van Leer limiter was used for the present work. A laminar numerical solution flow condition of Mach 10.6, wall to stagnation temperature ratio ($T_w/T_o$) of 0.30, and free-stream unit Reynolds number of $1.2 \times 10^6$ per foot for the first configuration was obtained. Figure 18 gives a comparison of measured data [Reference 31] and computed axial distribution of laminar heat-transfer for the first configuration. As seen in this figure, the computed rates of heat-transfer show excellent agreement with the experimental data. Unfortunately, there are no surface pressure and turbulent heat-transfer experimental data available for comparison for this model. Therefore, the second configuration was used for a better understanding of the salient features of the turbulent flowfield structure and determination of turbulent heat-transfer on axisymmetric blunt nosed bodies in hypersonic flow. The numerical solutions were generated at 5.0 and 10.6 free stream Mach number for the above mentioned second configuration. The wall to stagnation temperature ratio ($T_w/T_o$) used was 0.23 and 0.3 respectively. Comparison of measured data [Reference 32] and computed axial distribution of surface pressure and turbulent heat-transfer on the hemispherical 9° half-angle cone at a free stream Mach number of 5.0 and Reynolds number based on nose's diameter of $43.92 \times 10^6$ are given in Figures 19 and 20 respectively. Although the computed pressure results yield good agreement with the experimental data, the poor agreement between computed and measured turbulent heat-transfer results are noticeable. However, overall trends are clearly captured qualitatively. The flowfield about this geometry was also computed at free-stream Mach number of 10.6 and Reynolds number of $12.0 \times 10^6$ per foot. Figures 21 and 22 illustrate the computed surface pressure and turbulent heat-transfer comparison with experimental data for this configuration. Clearly shown in these figures is that

the computed surface pressure and turbulent heat–transfer are quantitatively and qualitatively in good agreement with the available experimental data.

## 2. BODY–STORE

The applicability and performance of the present flow solver was also tested for the flow over a generic two–dimensional planar body–store configuration at zero degree angle of incidence and free stream Mach number of 2.5 and Reynolds number based on body thickness width of $5.80 \times 10^6$, the viscous clustering yielded an average $y^+$ value of 4 for the first grid point off the body surface. Both body and store consist of an ogival forebody, a flat midsection and an ogival afterbody. Figure 23 displays this geometry. A H–type grid system is used to discretize the domain around the body–store configuration. The computational region for this configuration is divided into three blocks. Each block has a total of 5994 (81 x 37 x 2) points, for a sum total of 17982 points for the entire field grid. Steady state multiblock solutions are demonstrated by the flow about the body–store configuration with the store in captive position. Unsteady dynamic multiblock solutions are demonstrated by computing the flow of the complete multibody configuration as the store moves away from the parent body configuration through a prescribed vertical launch trajectory. The steady state calculation was started with initial conditions of free stream values everywhere, and then 2000 iterations of local time stepping with the CFL number equal to 3.5 was used to establish the steady state flow field before unsteady motion was started. The unsteady calculation starts after 2000 iterations of the steady state calculation. The plunge velocity of the store moving away from body was then suddenly set to a predetermined velocity and held constant thereafter; moreover, the local time step was replaced by a fixed time step, the same at all points, when the unsteady calculation began. The unsteady numerical solutions were continued until the store moved to a point four body widths away from the body. The overall pressure distribution of the body–store configuration as the store moves away from the body is depicted in Figure 24. Figure 24–a demonstrates the steady state computation with the store in captive position at the instant of vertical launch, which becomes the initial condition for the unsteady simulation. Moreover, Figure 24–a shows that the flow accelerates toward the shock wave on the lower flat portion of the body and then decelerates after crossing the shock; this interaction is strong enough to cause flow separation. Figures 24–b, c, d, e demonstrate the unsteady computation when the store is moving through the points of 0.5, 1.0, 1.5, and 2.0 body widths away from the body. In Figure 24 the viscous turbulent results are also compared with inviscid solutions at the same sequence of the store's position relative to the body. The evidence of viscous effect is mostly seen in the region between the store and body, particularly when the store is near the body. Note that the greatest difference in inviscid and viscous results is the prediction of the shock strength and position near the nose of the store. Shock strength is larger and shock position is farther downstream when the Euler model is used. Unfortunately, there are no experimental data available for comparison with the numerical results.

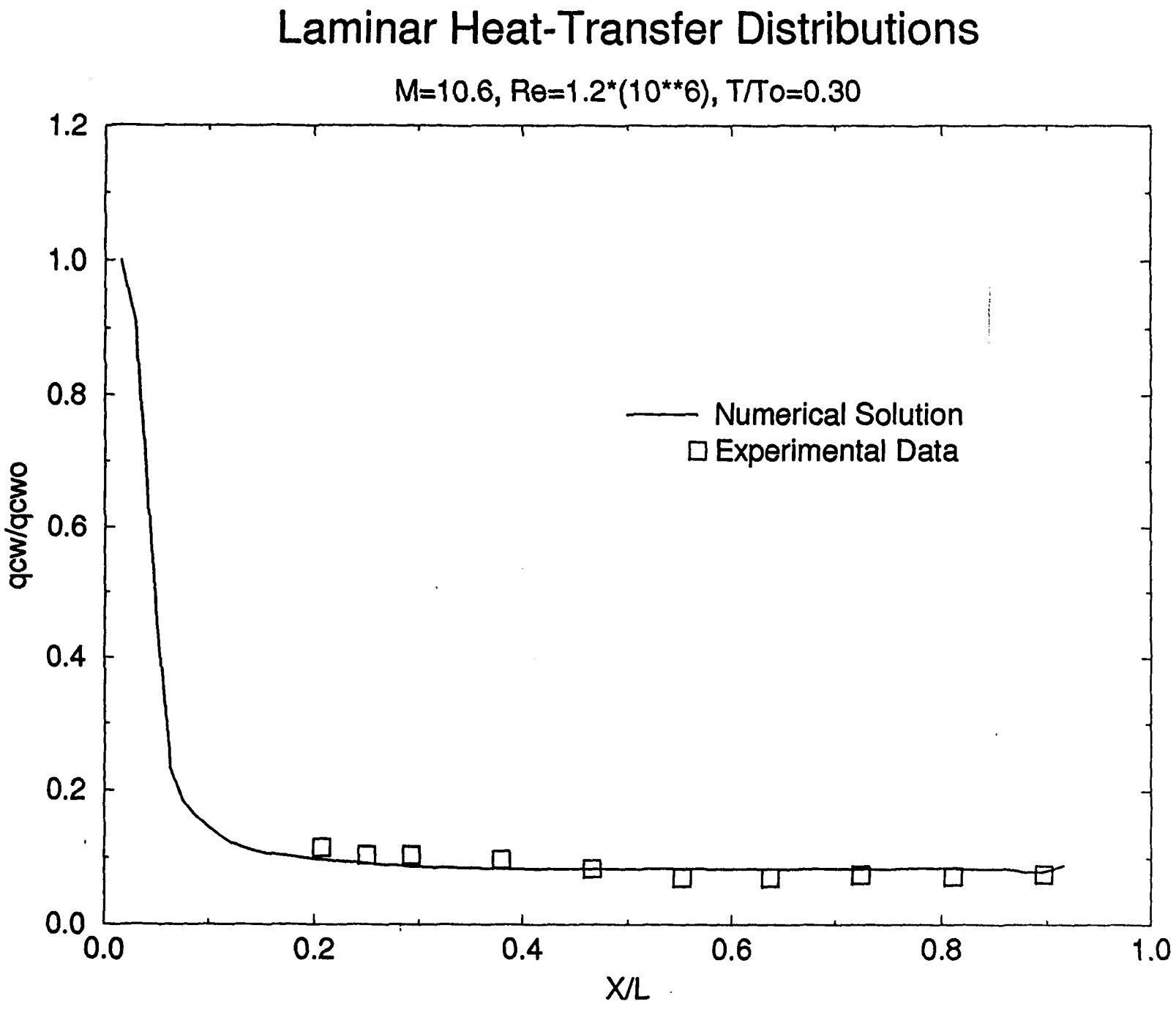



**Figure 18.** Computed and Measured Surface Heat-Transfer Distributions on Blunt (15°-semiapex spherical) Body at M = 10.6, $R_\infty$ = 1.2 x $10^6$

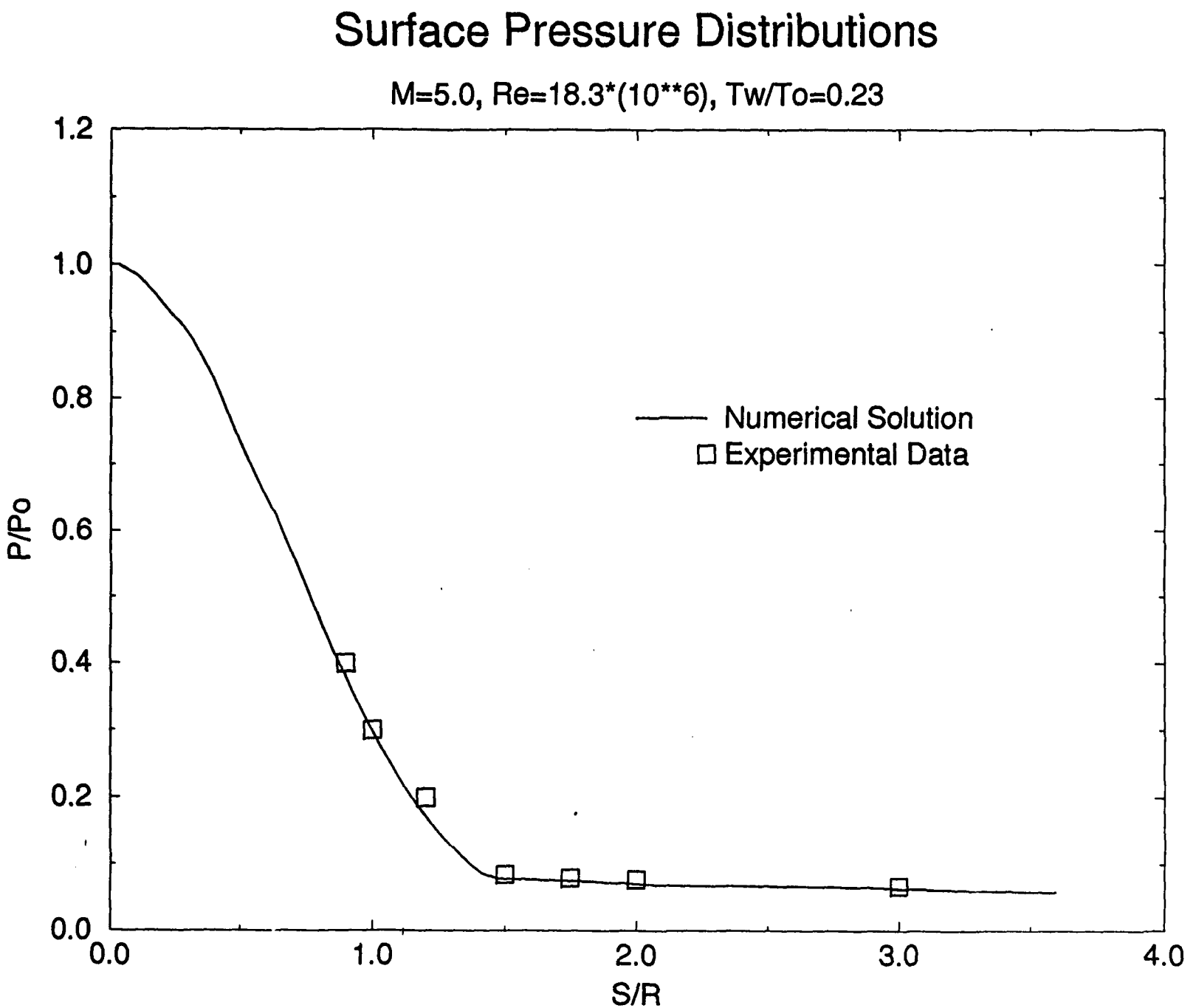



**Figure 19.** Computed and Measured Surface Pressure Distributions on Blunt (9° half-angle cone) Body at M = 5.0, $R_e$ = 18.3 x $10^6$

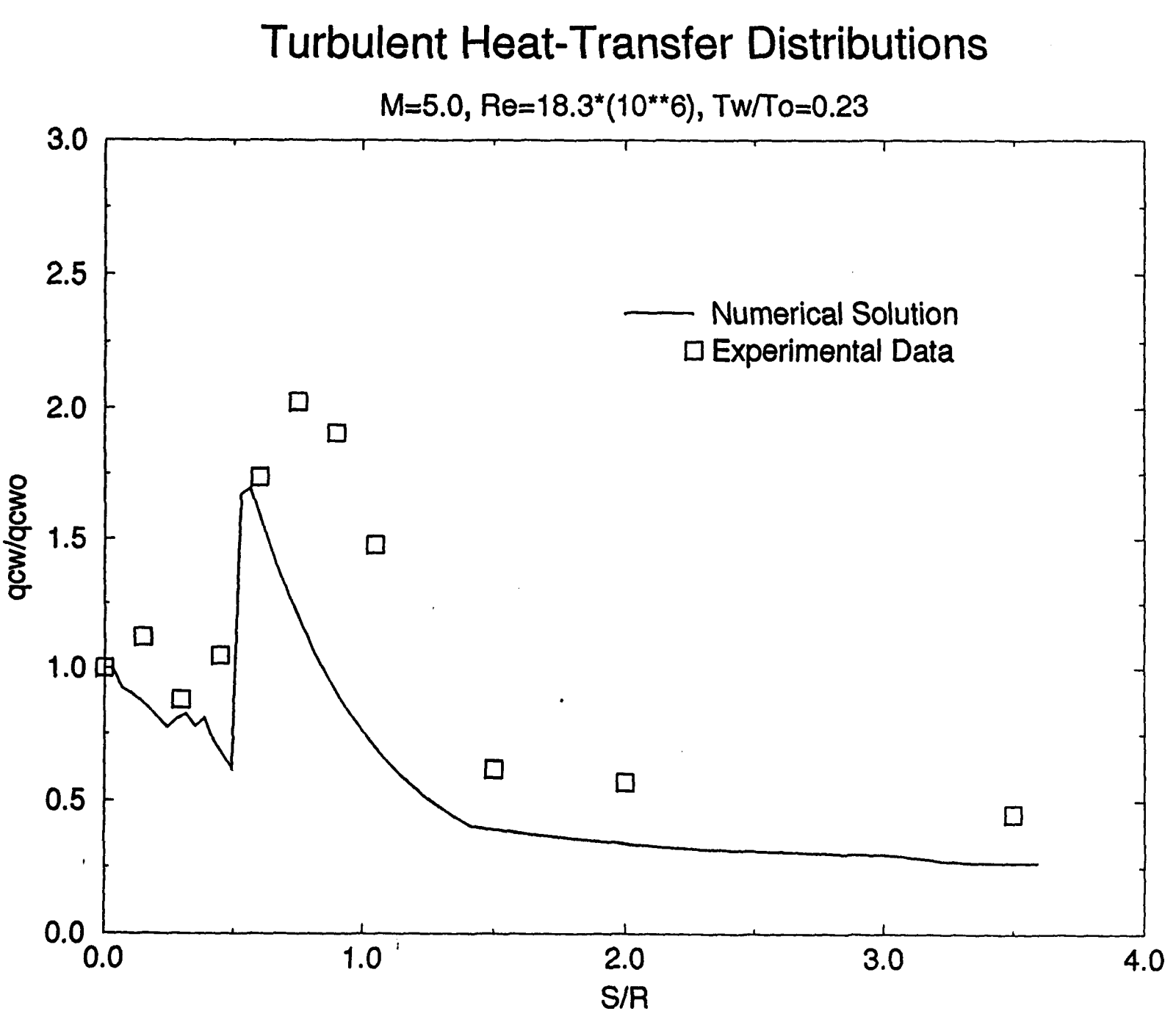



Figure 20. Computed and Measured Surface Heat-Transfer Distributions on Blunt (9° half-angle cone). Body at M = 5.0, $R_e$ = 18.3 x $10^6$

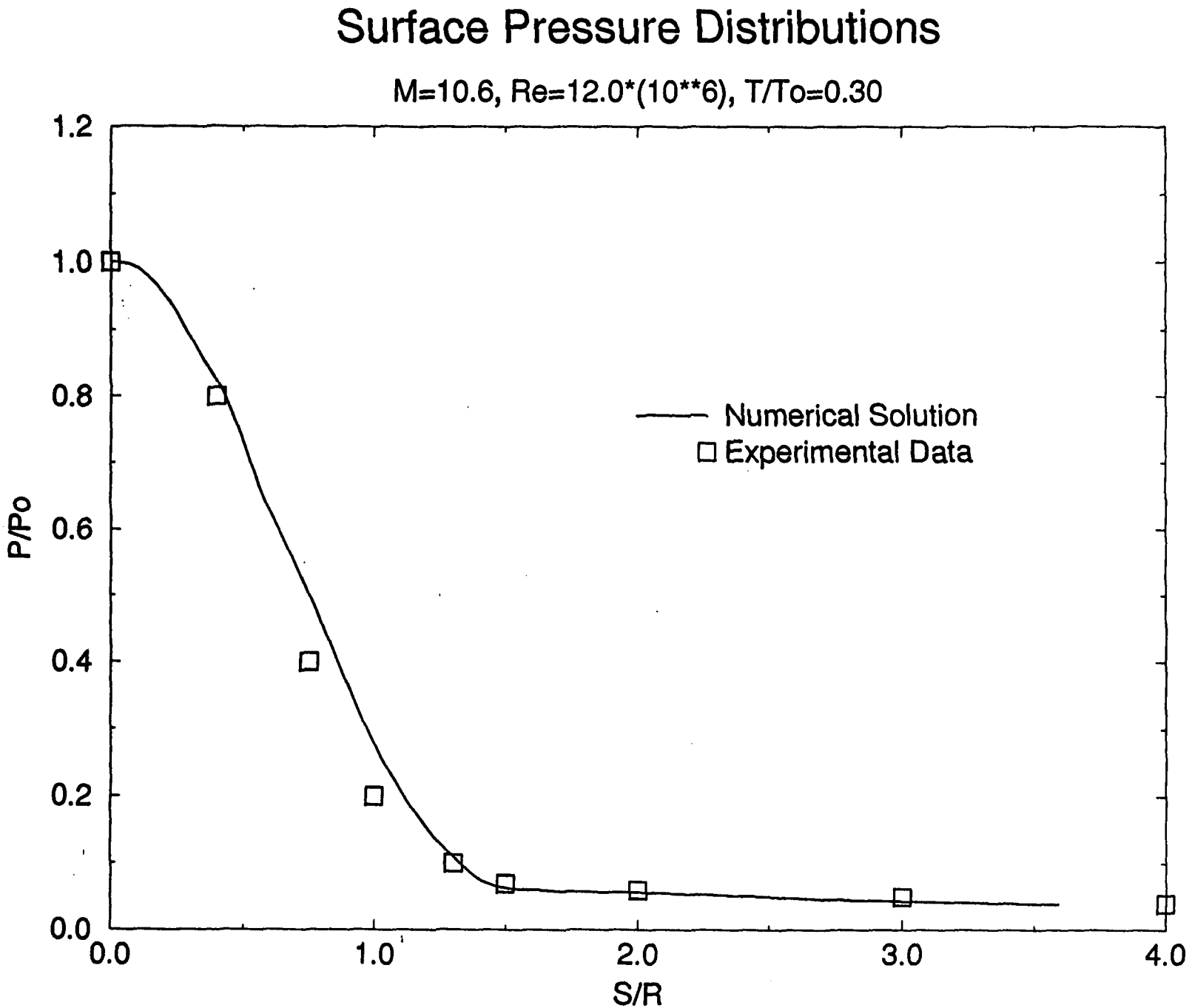



Figure 21. Computed and Measured Surface Pressure Distributions on Blunt (9° half-angle cone) Body at M = 10.6, $R_e$ = 12.0 x $10^6$

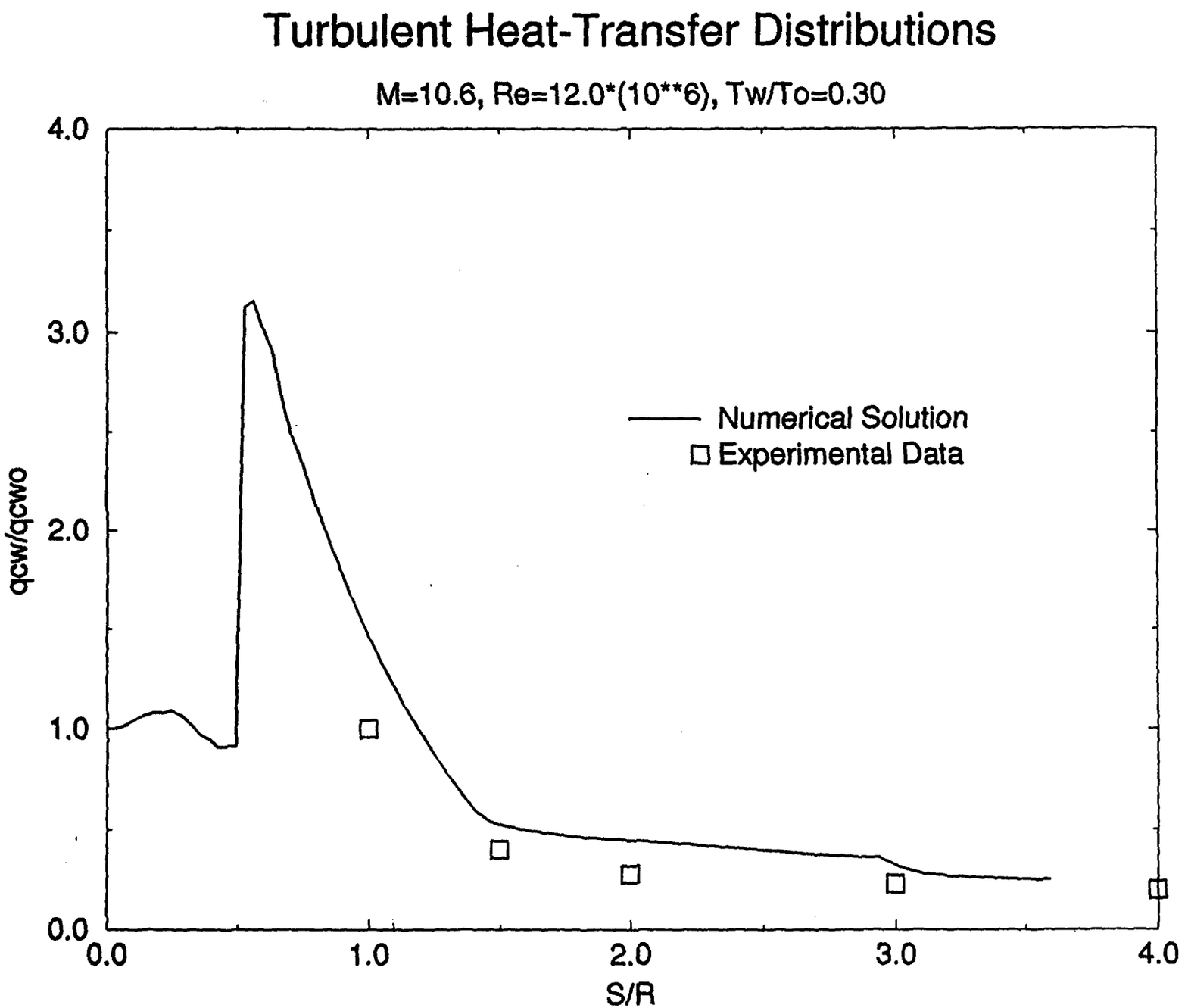



Figure 22. Computed and Measured Surface Heat-Transfer Distributions on Blunt (9° half-angle cone) Body at M = 10.6, $R_e$= 12.0 x $10^6$

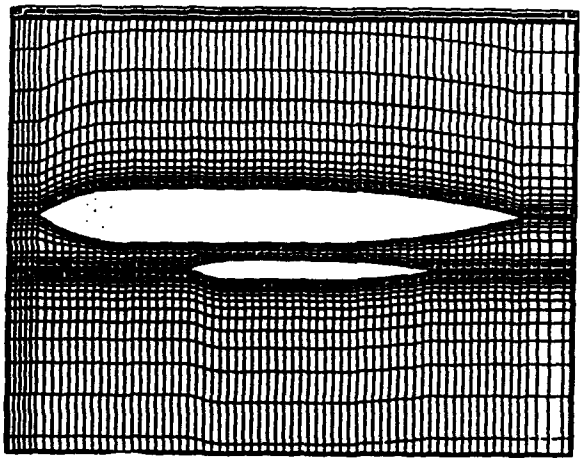


Figure 23. Perspective View of the Body-Store Configuration

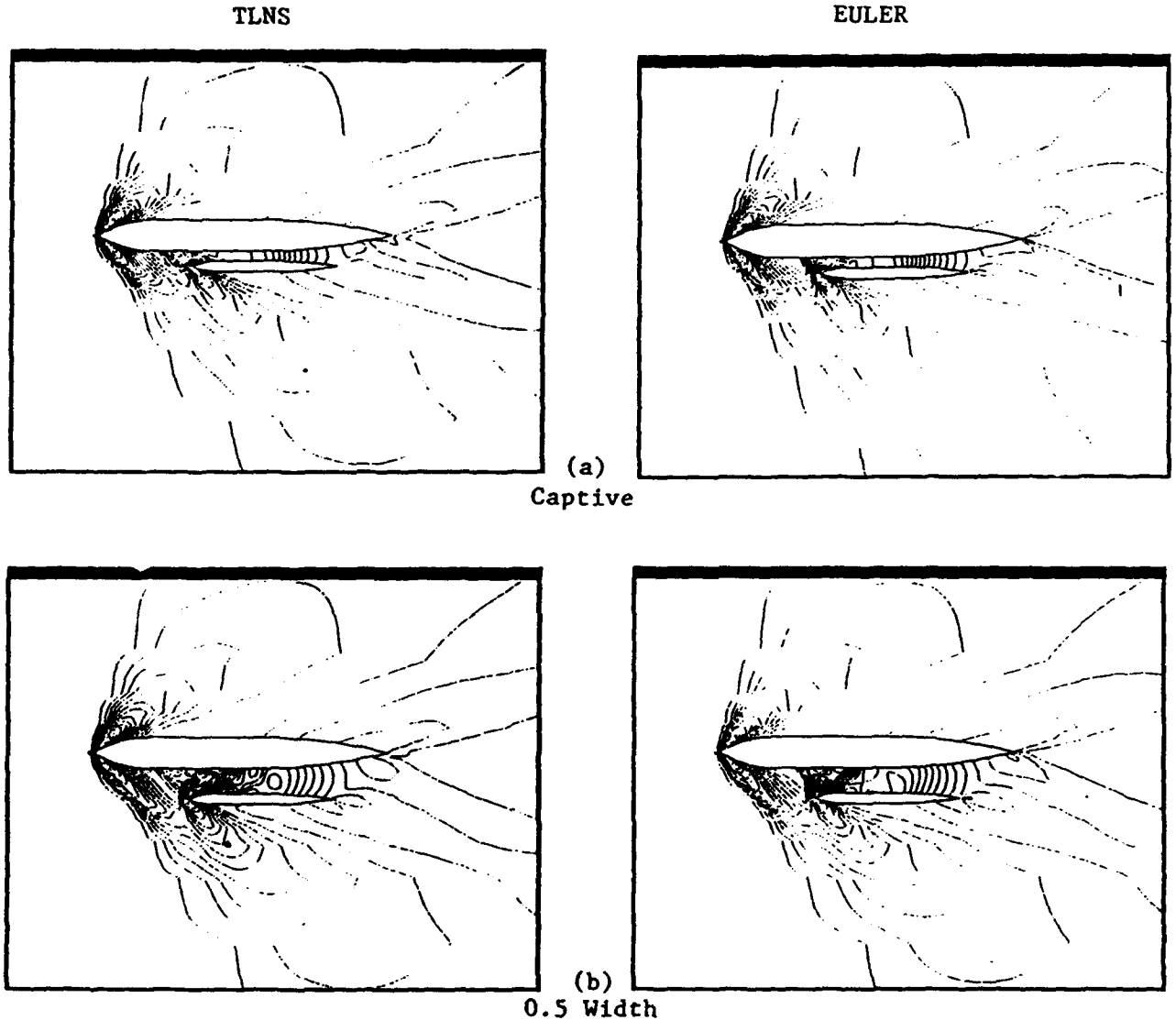
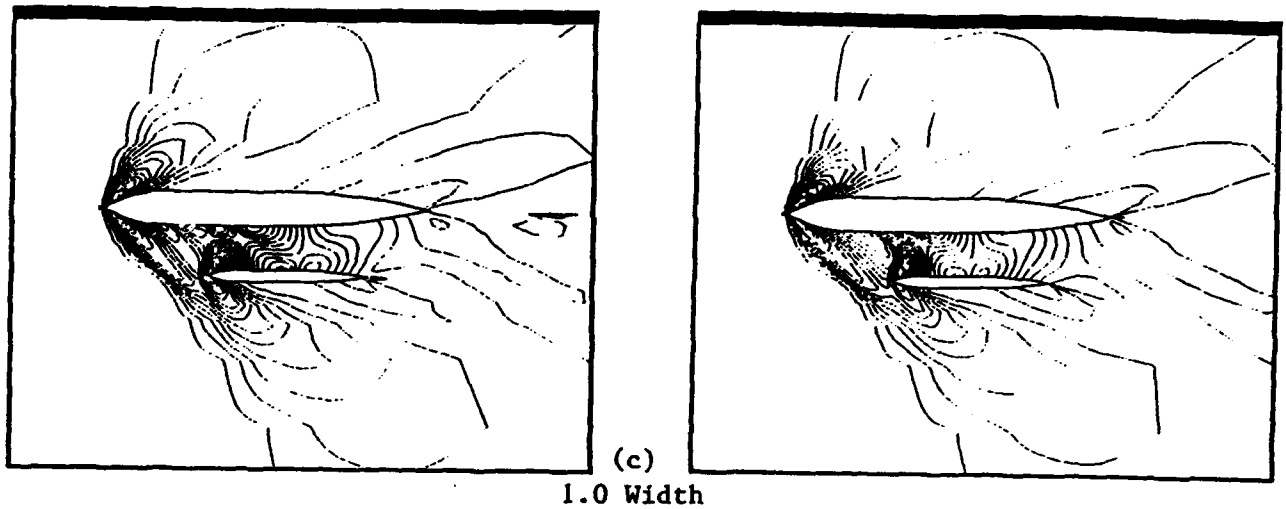
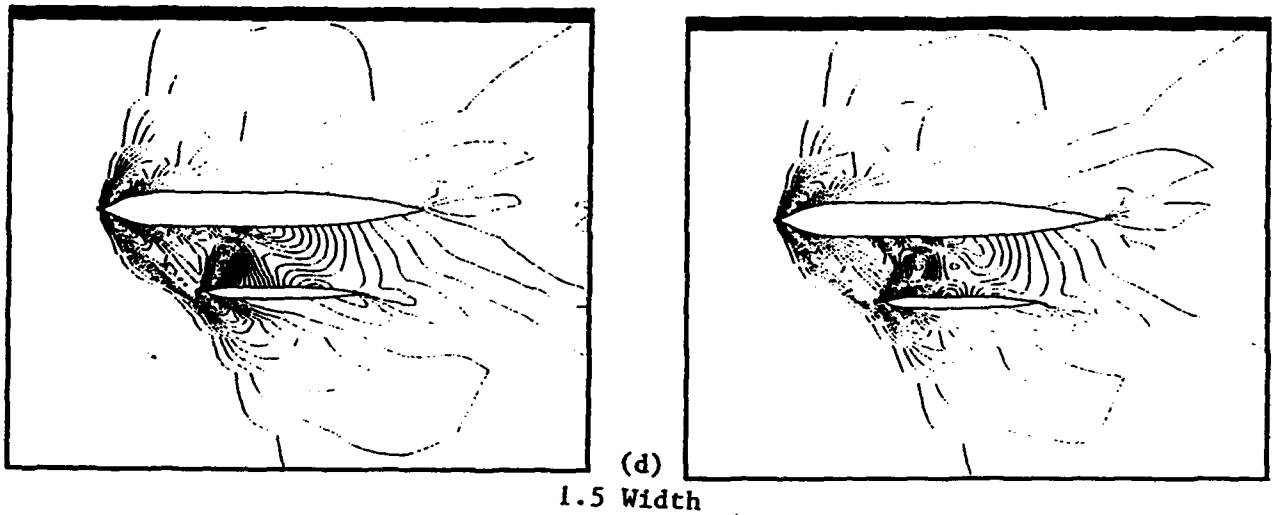



Figure 24. Computed Overall Pressure Distribution Contours with Store Located in Captive Position and Moving through 0.5, 1.0, 1.5, and 2.0 Body Widths Below the Body

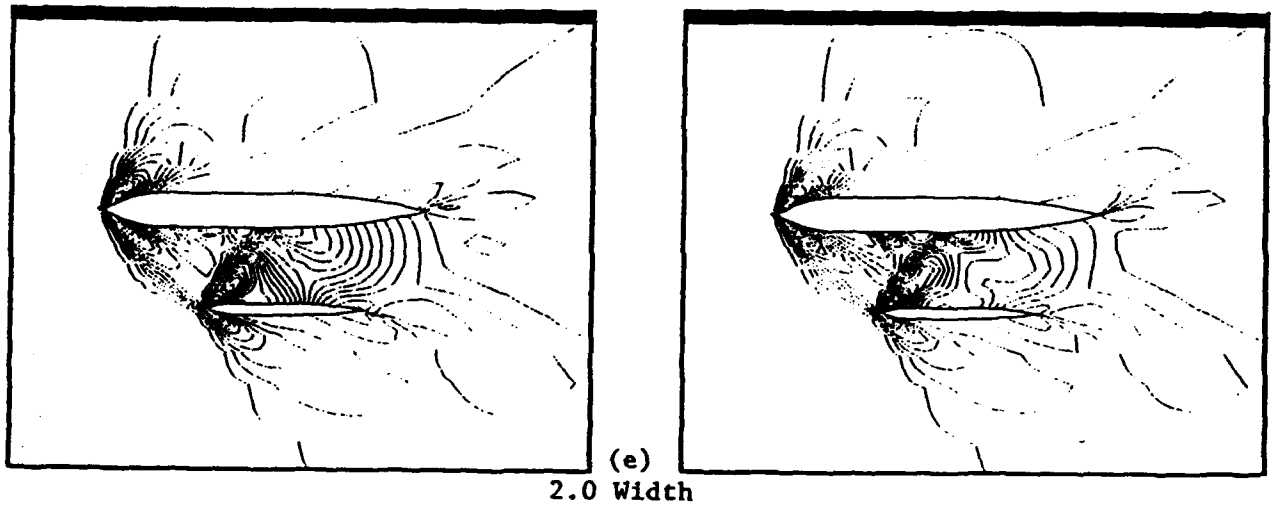
TLNS EULER



(c)
1.0 Width



(d)
1.5 Width



(e)
2.0 Width

Figure 24. Continued

# SECTION VII

# CONCLUDING REMARKS

Numerical methods for formulating and solving the unsteady three–dimensional compressible Euler and Navier–Stokes equations have been presented. In addition, an analysis of the modified two–pass numerical solution method has been performed and presented in terms of software optimization. The guidelines for the flow conditions of interest in this research was a freestream Mach number of 10 and below at an altitude of 100,000 feet and below. For such a flow regime the influence of chemistry can be important. As such there is also an additional investigation into the numerical formulation and solution of the equations including equilibrium chemistry. This effort is reported in a separate report as another phase of this same research effort.

## REFERENCES


1. J.L Steger, "Implicit Finite–Difference Simulation of Flow about Arbitrary Two–Dimensional Geometries," AIAA Journal, Vol. 16, No. 7, pp. 679–686, July 1978.

2. B. Gatlin, "An Implicit, Upwind Method for Obtaining Symbiotic Solutions to the Thin–Layer Navier–Stokes Equations," PhD Dissertation, Mississippi State University, August 1987.

3. L.B. Simpson, "Unsteady Three–Dimensional Thin–Layer Navier–Stokes Solutions on Dynamic Blocked Grids," PhD Dissertation, Mississippi State University, December 1988.

4. D.L. Whitfield and J.M. Janus, "Three Dimensional Unsteady Euler Equation Solutions Using Flux Vector Splitting," AIAA Paper No. 84–1552, June 1984.

5. D.M. Belk, "Three–Dimensional Euler Equations Solutions on Dynamic Blocked Grids," PhD Dissertation, Mississippi State University, August 1986.

6. B.S. Baldwin and H. Lomax, "Thin–Layer approximation and Algebraic Model for Separated Turbulent Flows," AIAA Paper No. 78–257, January 1978.

7. C. Hirsch, "Numerical Computation of Internal and External Flow, Volume 2: Computational Methods for Inviscid and Viscous Flows, John Wiley & Sons, New York, 1990.

8. L.B. Simpson and D.L. Whitfield, "Flux–Difference Split Algorithm for Unsteady Thin–Layer Navier–Stokes Solutions," AIAA Journal, Vol. 30, No. 4, pp. 914–922, April 1992.

9. P.L. Roe, "Approximate Riemann Solvers, Parameter Vector, and Difference Schemes," Journal of Computational Physics, Vol. 43, pp. 357–372, 1981.

10. J.L. Steger and R.F. Warming, "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Application to Finite Difference Methods," Journal of Computational Physics, Vol. 40, pp. 263–293, 1981.

11. B. Van Leer, "Towards the Ultimate Conservative Difference Scheme, V. A Second Order Sequel to Godunov's Method," Journal of Computational Physics, Vol. 32, pp. 101–136, 1979.

12. W.K. Anderson, J.L. Thomas, and B. Van Leer, "Comparison of Finite Volume Flux Vector Splitting for the Euler Equations," AIAA Journal, Vol. 24, pp. 1453–1460, 1986.

13. S. Osher and S. Chakravarthy, "Very High Order Accurate TVD Schemes," ICASE Report No. 84–44, September 1984.

14. D.L. Whitfield, "Implicit Upwind Finite Volume Scheme for the Three–Dimensional Euler Equations," Engineering and Industrial Research Station Report MSSU–EIRS–ASE–85–1, Mississippi State University, Mississippi State, MS, September 1985.

15. S.R. Chakravarthy and S. Osher, "A New Class of High Accuracy TVD Schemes for Hyperbolic Conservation Laws," AIAA Paper No. 85–0363, January 1985.

16. S.R. Chakravarthy and K.Y. Szema, "An Euler Solver for Three–Dimensional Supersonic Flows with Subsonic Pockets," AIAA Paper No. 85–1703, July 1985.

17. P.K. Sweby, "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws," SIAM Journal of Numerical Analysis, Vol. 21, No. 5, pp. 995–1011, October 1984.

18. J.M. Janus, "Advanced 3–D CFD Algorithm for Turbomachinery," PhD Dissertation, Mississippi State University, May 1989

19. D.L. Whitfield, "Implicit Upwind Finite Volume Scheme for the Three–Dimensional Euler Equations," Engineering and Industrial Research Station Report MSSU–EIRS–ASE–85–1, Mississippi State University, Mississippi State, MS, September 1985.

20. P.G. Buning and J.L. Steger, "Solution of the Two–Dimensional Euler Equations with Generalized Coordinate Transformation Using Flux Vector Splitting," AIAA Paper No. 82–0971, June 1982.

21. J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.

22. J.E. Dennis Jr. and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice–Hall Inc., Englewood Cliffs, New Jersey, 1983.

23. W.R. Briley and H. McDonald, "Solution of the Multidimensional Compressible Navier–Stokes Equations by a Generalized Implicit Method," Journal of Computational Physics, Vol. 24, pp. 372–397, 1977.

24. R.M. Beam and R.F. Warming, "An Implicit Finite–Difference Algorithm for Hyperbolic Systems in Conservation Law Form," Journal of Computational Physics, Vol. 22, pp. 87–110, 1976.

25. W.K. Anderson, "Implicit Multigrid Algorithms for the Three–Dimensional Euler Equations," PhD Dissertation, Mississippi State University, Mississippi State, MS, August 1986.

26. F.A. Mansfield, "Investigation of Solution Operators for the Three–Dimensional Unsteady Euler Equations," PhD Dissertation, Mississippi State University, Mississippi State, MS, May 1991.

27. A.A. Samarskii and E.S. Nikolaev, Numerical Methods for Grid Equations, Volume II, Iterative Methods, Birkhauser Verlag, Boston, 1989.

28. T.J. Barth, "Analysis of Implicit Local Linearizations Techniques for Upwind and TVD Algorithms," AIAA Paper No. 87–0595, January 1987.

29. D.L. Whitfield and L.K. Taylor, "Discretized Newton–Relaxation Solution of High Resolution Flux–Difference Split Schemes," AIAA Paper No. 91–1539, June 1991.

30. Cray X–MP and Cray –1 Computer Systems (Fortran (CFT) Reference Manual SR–0009), CRI, Mendota Heights, Minnesota, 1984.

31. J.W. Cleary "Effects of Angle of Attack and Bluntness on Laminar Heating–Rate Distributions of a 15° Cone at a Mach Number of 10.6," NASA TN D–5450, October 1969.

32. G.F. Widhope and R. Hall, "Transitional and Turbulent Heat–Transfer Measurements on a Yawed Blunt Conical Nosetip," AIAA Journal, Vol. 10, No. 10, October 1972, pp. 1318–1325.

DISTRIBUTION
(WL-TR-92-7044)

Defense Technical Info. Center
Attn: DTIC/DDAC
Cameron Station — 2
Alexandria VA 22304-6145

AFSAA/SAI
The Pentagon, Rm 1D363 — 1
Washington DC 20330-5420

AUL/LSE — 1
Maxwell AFB AL 36112-5564

Eglin AFB offices:

WL/CA-N — 1
WL/MNOI (Scientific and Tech. Info. Facility) — 1

HQ USAFE/INATW — 1
APO NY 09012-5001

WL/FIES/SURVIAC — 1
Wright-Patterson AFB OH 45433-6553

---

HQ ACC/XP-JSG — 1
Langley AFB VA 23665-5000

Eglin AFB offices:

ASC/XRC — 1
WL/MNSI — 1
WL/MNAG — 1
WL/MNM — 1
WL/MNAV — 1
WL/MNPX — 1
WL/MNAA — 4
AFDTC/PA — 1

Commander — 1
U.S. Army Missile Command
Redstone Sci. Info. Center
Attn: AMSMI-RD-CS-R/Documents
Redstone Arsenal AL 35898-5241

Commander — 1
Naval Weapons Center (Code 3431)
Attn: Technical Library
China Lake CA 93555-6001

ASC/ENSTA — 1
Wright-Patterson AFB OH 45433-6503

ASC/XRH — 1
Wright-Patterson AFB OH 45433-6503

Wright-Patterson AFB OH 45433-6553

WL/CA-F — 1
WL/FIM — 1
WL/FIB — 1
WL/FIGX — 1
WL/FIGCC — 1

WL/TXA — 1
Wright-Patterson AFB OH 45433-6523

AFIA/INT — 1
Bolling AFB DC 20332-5000

EOARD/LDV
Box 14 — 1
FPO NY 09510-0200

NASA Langley Research Center — 1
Technical Library - MS 185
Attn: Document Cataloging
Hampton VA 23665-5225